4
Arquitetura para Adaptação de Conteúdo em Sistemas Publish/Subscribe

A crescente popularidade de serviços de informação que demandam entrega de informações (notícias, da bolsa de valores, etc) para clientes móveis motiva a necessidade por um serviço eficiente e escalável de disseminação de informação que permita a entrega de conteúdo personalizado a usuários móveis.

Middlewares publish/subscribe (Pub/Sub) permitem que assinantes definam as características do conteúdo de interesse, e sejam notificados quando tal conteúdo se torna disponível. Alguns sistemas publish/subscribe com algoritmos eficientes de roteamento resolvem questões relacionadas à mobilidade dos clientes [63, 29]. Entretanto, a diversidade de demandas e a natureza dinâmica dos ambientes móveis requerem serviços adicionais à funcionalidade de difusão de mensagens do middleware Pub/Sub para oferecer flexibilidade e customização em relação ao contexto particular de cada usuário.

Este capítulo lista os principais requisitos de um serviço de disseminação de conteúdo customizado para usuários móveis (Seção 4.1) e apresenta uma arquitetura de referência para este tipo de serviço. As características dos componentes da arquitetura, bem como sua interação, são descritos na Seção 4.2. Na Seção 4.3, são apresentados algoritmos para melhorar a escalabilidade do sistema que identificam clientes com contextos comuns, e permitem a customização eficiente de conteúdo.

4.1
Requisitos

Um requisito comum para uma aplicação para ambientes móveis é que ela seja capaz de entregar conteúdo altamente personalizado e customizado, de acordo com as preferências e contexto atual dos usuários. O foco principal da arquitetura proposta são serviços de disseminação de conteúdos, no qual os provedores de conteúdo e clientes são conectados via comunicação publish/subscribe. Desta forma, os clientes atuam principalmente como assinantes do serviço, e servidores de conteúdo atuam como os publicadores de conteúdo não-personalizado. Há portanto a necessidade de um proxy (entre

publicadores e assinantes de conteúdo) que seja responsável por filtrar e transformar a informação de acordo com as necessidades de cada usuário. Os seguintes requisitos deveriam ser atendidos pelo serviço:

Comunicação tipo *push*. Os usuários deste tipo de serviço devem ser capazes de definir o tipo de conteúdo que desejam receber, e devem receber a informação publicada assim que ela estiver disponível. A entrega do conteúdo, seguindo a abordagem *push*, elimina a sobrecarga de requisições explícitas por parte dos clientes em intervalos regulares, e é mais adequada à natureza não-determinística da criação e publicação de conteúdo.

Customização de conteúdo. O sistema deve permitir a customização e a adaptação do conteúdo publicado ao contexto do usuário. Por exemplo, pode-se remover imagens e manter só o texto de uma página HTML caso a conectividade do dispositivo esteja muito ruim, ou pode-se redimensionar imagens ao tamanho de tela do dispositivo. A customização reduz a sobrecarga de recepção e armazenamento de informação indesejada no usuário, bem como permite a seleção de cada parte de informação publicada adaptando-a ao contexto e às preferências do usuário.

Tratamento de desconexão. O sistema deve prover tratamento a períodos de desconexão, permitindo definir diferentes políticas para o armazenamento temporário das mensagens publicadas para clientes desconectados [1].

Escalabilidade. O serviço deve ser capaz de lidar com um grande número usuários potenciais, e deve ser otimizado para a área de aplicação particular com respeito ao número de publicadores e de assinantes no sistema, ao tamanho e à freqüência do conteúdo publicado.

Extensibilidade. O serviço deve permitir incorporar novas regras de customização e bem como criar adaptações mais apropriadas ao tipo de conteúdo publicado, de forma simples e de preferência dinamicamente (isto é, sem interrupção do serviço de disseminação).

Transparência. A gerência da adaptação deve ser transparente aos clientes, não sendo necessário que eles se preocupem como as adaptações são selecionadas (critérios) e onde são efetuadas.

[1] Neste trabalho, o foco é o tratamento de desconexão para a comunicação no sentido servidor (publicador) para cliente. Está fora do escopo a manutenção da operação no cliente em períodos de desconexão (ou fraca conectividade), como é realizado em sistemas como CODA e Odyssey [107]

4.2 Arquitetura Proposta

Os requisitos descritos na seção anterior nortearam o projeto de uma arquitetura de referência para serviços de disseminação de informações para clientes móveis (aplicações) sobre sistemas publish/subscribe, e que provê adaptação individualizada segundo o contexto de cada cliente. A Figura 4.1 ilustra a arquitetura proposta.

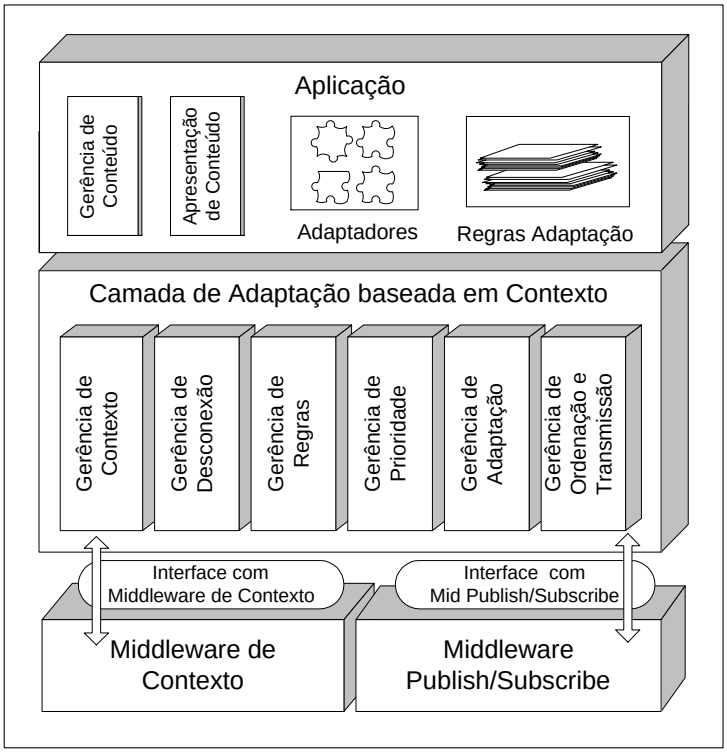

Figura 4.1: Arquitetura para sistemas Pub/Sub com adaptação sensível a contexto

A arquitetura é formada logicamente por três camadas:

i) camada base - composta pelo middleware de comunicação Publish/Subscribe e pelo middleware de provisão de contexto. O middleware Pub/Sub é responsável pela intermediação da comunicação entre servidores (de conteúdo) e clientes, e pode ser utilizado por outros componentes que requeiram este tipo de comunicação. Ele deve também tratar questões de mobilidade dos usuários, tais como roteamento e evitar perda de mensagens devido a *handoff* ou indisponibilidade de clientes. O Middleware de provisão de contexto é responsável por coletar e inferir (situações de) contextos de interesse da aplicação, informando a ocorrência destes para permitir a execução de adaptações adequadas.

ii) camada intermediária - composta de uma série de gerentes que são necessários para a disseminação customizada de conteúdo, através da aplicação de adaptações baseadas no contexto corrente dos clientes.

iii) camada de aplicação - compostas por elementos que tratam as necesidades específicas da aplicação, tais como, a gerência de conteúdo (nos servidores), a forma de apresentação de conteúdo (nos clientes), e definição das situações de contexto que são de interesse para a aplicação e a definição das adaptações adequadas em cada situação.

Essa divisão em camadas visa a separação de responsabilidades. Na camada base, são tratadas as questões de comunicação, tanto entre servidores de conteúdo e clientes (middleware Pub/Sub), quanto com o sistema de provisão de contexto. A camada intermediária concentra as funções comuns (a quaisquer sistemas com adaptação) necessárias para a gerência de adaptações baseada em contexto, lidando com as questões de aquisição de contextos e associação destes com as adaptações apropriadas. Na camada de aplicação, estão elementos específicos da aplicação (ou serviço de disseminação de conteúdo), como a forma de apresentação nos clientes, e as situações de contexto que deseja tratar e como tratá-las (adaptações). Essas camadas (e seus componentes) podem estar distribuídas em diversos nós da rede.

A arquitetura proposta foi projetada de modo a permitir que desenvolvedores de aplicações móveis se concentrem na *lógica de negócio*, sem precisar lidar com questões específicas inerentes aos ambientes móveis, como por exemplo, gerenciamento das condições do ambiente (contextos), a ativação e execução de adaptações, o gerenciamento de desconexões, etc. O desenvolvedor ou administrador da aplicação precisa apenas definir as regras de adaptação. Os componentes [2] da arquitetura são descritos com mais detalhes nas próximas seções.

4.2.1 Middleware de comunicação Publish/Subscribe

O modelo de comunicação publish/subscribe é a base para a interação entre os elementos na arquitetura proposta. Fica a cargo do middleware Pub/Sub o armazenamento e a distribuição das mensagens entre produtores e consumidores. Alguns sistemas fornecem persistência das mensagens, permitindo que um cliente que não esteja ativo no momento possa recebê-los posteriormente.

[2] O termo componente está sendo usado no sentido amplo, e não com o significado de "componente de software", definido em [108] como "uma unidade de composição com interfaces contratualmente especificadas e somente com dependências contextuais explícitas".

A arquitetura propõe a utilização de um middleware Pub/Sub como infra-estrutura de comunicação. Este middleware é utilizado para disseminação de conteúdo entre produtores e consumidores (clientes), e também serve como canal de distribuição dos eventos de contexto (fornecidos pelo middleware de contexto) que controlam a adaptação. O middleware Pub/Sub deve permitir que consumidores registrem interesse (através de *subscriptions*) em conteúdos publicados, segundo um conjunto de restrições. O esquema de registro de interesse utilizado pode ser baseado em tópico, em conteúdo, ou outro esquema qualquer [52], como descrito na Seção 2.3.

O middleware é responsável por oferecer suporte à mobilidade dos clientes, com gerência de *handover* [3] (horizontal em nível de rede) e técnicas de predição de mobilidade. A gerência de *handover* envolve etapas de: detecção de início *handover*, estabelecimento de nova conexão e redirecionamento do fluxo de dados. As técnicas de predição de mobilidade auxiliam na identificação do *handover* e previsão/manutenção de QoS [109, 110]. Outra característica desejável é que o middleware suporte *handover* vertical [111, 112], no qual clientes podem chavear entre enlaces com diferentes tecnologias e protocolos de transporte (Wifi, GPRS, SMS, etc). Alguns middlewares Pub/Sub que provêm suporte à mobilidade são: Siena [54], Elvin [56] e JEDI [57]. Além disso, é interessante que o sistema Pub/Sub tenha uma arquitetura distribuída para lidar com os problemas relacionados à escalabilidade [28].

Os sistemas Pub/Sub oferecem um conjunto proprietário de Interfaces de Programação de Aplicações (APIs) para a integração com outros sistemas. Por isso se faz necessário o componente Interface Middleware Pub/Sub, que atua como um mediador entre a camada de serviços de adaptação e o middleware Pub/Sub. Para a interação com a camada de gerência de adaptações, o middleware Pub/Sub deve prover meios para a interceptação de notificações de publicações. Duas funcionalidades principais devem ser permitidas: *i*) a recuperação do conjunto de clientes destinatários de uma notificação, e *ii*) a publicação para subconjuntos de clientes, com alteração do conteúdo da notificação (Figura 4.2). A primeira é necessária para que sejam identificados os clientes destinatários da notificação, para que então possa se avaliar as informações de contexto de cada um e definir as adaptações correspondentes. A segunda interface deve permitir que o conjunto de destinatários da notificação seja modificado para subgrupos do conjunto inicial, e, para cada subconjunto, um novo conteúdo da notificação seja encaminhado (*sub-notificação*). Este novo conteúdo consiste do conteúdo original modificado pelas adaptações

[3] Mecanismos necessários para manutenção da conexão e continuidade de serviços durante a migração do dispositivo móvel entre células ou pontos de acesso

adequadas às condições de contexto dos clientes.

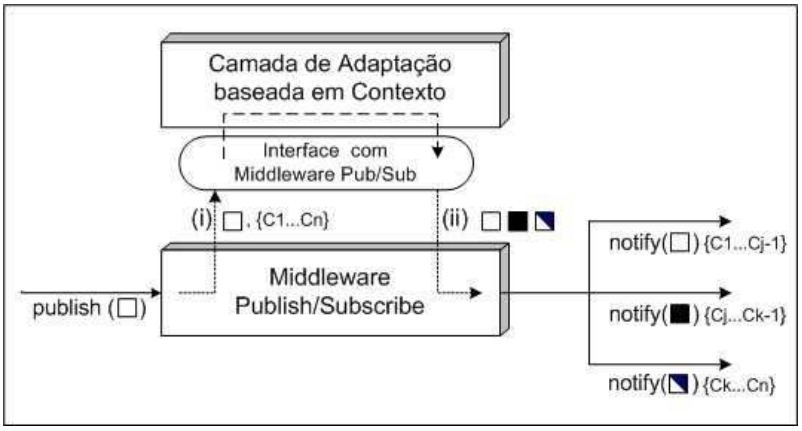

Figura 4.2: Interação com middleware Pub/Sub

4.2.2 Middleware de provisão de contexto

O middleware de provisão de contexto é responsável por coletar, inferir e distribuir as informações de contexto dos clientes que sejam de interesse da aplicação ou serviço. Ele se comunica com a camada de adaptação através de eventos que informam a ocorrência de uma situação de contexto, utilizada para a ativação da adaptação correspondente, e de eventos indicando quando uma dada situação de contexto deixa de existir, evento que é necessário para desativação da adaptação. Ele também é responsável por informar uma possível desconexão (permanente ou temporária) de dispositivos, e sua posterior reconexão.

A maioria dos sistemas utiliza apenas contextos primitivos, tais como perfis de usuários, de dispositivos e dados sobre a capacidade do enlace sem fio, para determinar a adaptação de conteúdo necessária. Entretanto, a arquitetura proposta sugere que o middleware de provisão de contexto utilizado permitia a definição de contextos complexos, e não apenas os contextos primitivos, para melhor se aproximar às abstrações e necessidades da aplicação, na definição das situações de contexto de interesse. Define-se como **contexto complexo** (ou inferido), os contextos que são agregados ou derivados de outras informações primitivas. A localização (derivada de diversas leituras de sensores) e velocidade de movimentação (localização em conjunto com tempo) são exemplos de contextos complexos, e que podem fornecer informações de interesse para adaptação. Para exemplificar a necessidade desse tipo de contexto, três cenários de aplicações que podem usar contexto complexo para ativar adaptações são mostrados a seguir. Uma aplicação de mapa pode,

devido à pequena tela de um dispositivo e tendo a localização corrente do usuário como ponto de referência, adaptar a imagem fazendo um zoom ou um recorte (*clipping*) na região do mapa em torno da posição do usuário. Outra adaptação possível é a restrição (ou bloqueio) de conteúdo de acordo com a localização, ou seja, limitar o acesso a transparências eletrônicas, vídeos ou documentos, apenas às pessoas localizadas nos prédios de uma empresa, ou sala de conferências, enquanto que pessoas fora desta área não teriam acesso a tais documentos, mas apenas receberiam um resumo dos documentos. Um terceiro exemplo é o uso da informação de contexto "está em movimento", ou seja, situação em que um cliente está mudando rapidamente de uma região para outra. Uma aplicação poderia, então, enviar aos clientes nesta situação ("em movimento") apenas um resumo da informação, e esta, formatada em letras maiores, ou mandar somente uma notificação de chegada de mensagens, podendo armazená-las para posterior visualização. Este tipo de situação e os exemplos anteriores mostram a necessidade de uso de contexto complexo (por exemplo, inferência se a posição corrente está dentro dos limites de um prédio) e não apenas de dados simples sobre a rede ou dispositivo como contexto.

A função do middleware inclui interpretar o contexto baseado nos dados coletados, agregando informação contextual proveniente de múltiplas fontes (perfis , banco de dados, sensores, serviços), detectando e resolvendo as inconsistências das informações. Portanto, deve prover meios para a composição das mesmas, e tratar problemas inerentes a fontes distribuídas, como por exemplo, ordenação de ocorrência de eventos mesmo que estes cheguem ao sistema (dentro de uma janela de tempo configurável) em fora de ordem, por exemplo, devido a retardos na transmissão.

É desejável que o middleware de contexto trate de questões como segurança e privacidade de informações de contexto [6, 113, 114], permitindo-se definir um conjunto de restrições envolvendo preferências de privacidade dos usuários e as políticas de privacidade das aplicações, permitindo estabelecer limites de visibilidade para os dados coletados.

Outros aspectos importantes incluem flutuações e instabilidade de contexto [115, 116]. Tais flutuações são específicas a cada tipo de contexto, e dependem da forma como os dados são coletados. A oscilação do contexto pode tornar-se um grave problema para aplicações adaptativas, já que uma variação no contexto pode ativar uma nova cadeia de adaptações, que pode ser aplicada desnecessariamente, levando a um aumento no consumo de recursos, e possivelmente até causando adaptações de conteúdo inapropriadas. Algumas possíveis soluções incluem: a incorporação de filtros associados a alguns tipos de contexto para suavizar variações (como RSSI-GreyModel [109], para

qualidade/variação de sinal RSSI); uso de intervalos na definição de valores de interesse no contexto; formas diferentes de reação à mudança de contexto, imediata (para desconexão) ou reação atrasada (lazy), de acordo com o tipo de contexto.

O componente de interface entre o middleware de provisão de contexto e a camada de adaptação define como o contexto deve ser requisitado e as possíveis formas de composição. A definição do contexto de interesse depende da modelagem de contexto adotada pelo middleware. As principais abordagens para a definição e modelagem de contexto, segundo [40, 41], são: modelos de tuplas objeto-valor [38], modelo de esquema de linguagens Mark-up [44], modelos orientados a objetos [47], modelos baseados em lógica e modelos baseados em ontologias [50]. Dependendo da modelagem utilizada, a definição das regras de adaptação (os contextos de interesse) pode se tornar mais (ou menos) simples, e significativa. Em geral, eventos de contextos complexos são definidos por uma expressão de contexto, na qual se combinam os atributos de contexto através de operadores lógicos de conjunção, disjunção e negação. Além disso, para cada atributo de contexto, pode haver restrições definidas por operadores relacionais, como igual, diferente, maior ou menor, etc. Pode haver também uma representação XML para a definição das meta-informações sobre cada atributo de contexto, como por exemplo, a fonte, sensor ou serviço, o grau de precisão, intervalos de confiança do valor da fonte, etc. Essas expressões representam as condições nas quais deverão ser aplicadas as adaptações.

4.2.3 Camada de aplicação

A camada de aplicação contém componentes que são específicos à aplicação de disseminação de conteúdo, sendo afetada pelo tipo de conteúdo distribuído e pelo propósito particular da aplicação.

O componente *gerência de conteúdo* permite o(s) publicador(es) definir os canais ou tópicos de publicação e gerencia o conteúdo que é publicado nos diferentes canais. Este elemento é também responsável por disponibilizar meta-informações sobre o conteúdo disseminado, tais como título, descrição, língua, tipo e formato de dado, resolução, etc. Essas informações podem ter uma representação simples, por exemplo em XML, e são importantes para auxiliar o processo de definição das adaptações a serem utilizadas. Por exemplo, pode-se verificar se um adaptador é apropriado (ou é capaz de adaptar) ao conteúdo, avaliando seu tipo e formato.

O componente *apresentação de conteúdo* é responsável pela apresentação de conteúdo em diferentes dispositivos. Pode-se usar XML e tecnologias rela-

cionadas para criar e gerenciar interfaces flexíveis [117], e estudos sobre outros problemas relacionados à apresentação, como estruturação e particionamento de conteúdo.

As *regras de adaptação* definem as condições nas quais devem ser aplicadas as adaptações de conteúdo. Desta forma, elas relacionam as situações de contextos de interesse com as adaptações adequadas para cada situação. Estas regras são específicas para a aplicação e devem ser definidas/configuradas pelo desenvolvedor da aplicação ou pelo administrador do sistema. Também podem ser definidas regras para momentos de desconexão que ativam o armazenamento de conteúdo.

A camada de aplicação é responsável por definir e implementar as adaptações para atender às necessidades específicas da aplicação. Os *adaptadores* de conteúdo (por exemplo, para conversão de formatos, como HTML para WML, EPS para JPEG, ou redimensionamento de imagens) também podem ser desenvolvidos por terceiros e estarem armazenados em um ou mais repositórios, formando bibliotecas de adaptadores.

Os adaptadores devem ter perfis ou descritores com informações sobre o tipo de conteúdo que ele adapta, o conteúdo resultante, a interface de invocação, descrição da transformação, custo, tempo de execução, etc. Essas informações semânticas são importantes para a descoberta e a composição de serviços de adaptação (de forma automática)

4.2.4 Camada de gerência de adaptação baseada em contexto

A camada de adaptação é responsável por realizar as adaptações do conteúdo de forma adequada a cada cliente. Ela relaciona os contextos de interesse da aplicação com as adaptações desejadas, através das regras de adaptação definidas pela aplicação, e então, ativa/desativa as adaptações quando necessário. As notificações do middleware Pub/Sub são interceptadas, e o conteúdo é modificado de acordo com o contexto dos clientes, e por fim, é retransmitido pelo middleware Pub/Sub.

Esta camada intermediária deve estar implementada em nós da rede fixa, mas é interessante que esteja próxima aos clientes, pois as adaptações visam otimizar principalmente os recursos do enlace sem fio dos dispositivos. Ela pode estar distribuída em diversos nós para melhor eficiência. Uma boa opção é acoplá-la ao(s) broker(s) do middleware Pub/Sub, responsáveis pela distribuição das notificações (discutido na Seção 2.3). Neste caso, a camada deve poder interceptar o envio das notificações para realizar as modificações necessárias no conteúdo e nos destinatários. Outra opção é transformá-la em

um proxy intermediário entre o sistema Pub/Sub e os clientes. Neste caso, o proxy intermediaria a comunicação com os clientes, atuando como um "servidor"para eles, e como um cliente para o middleware Pub/Sub.

A camada de adaptação é composta por uma série de componentes que são detalhados a seguir. A idéia desta decomposição surgiu de experiências práticas de implementação de aplicações de adaptação conteúdo baseada em contexto, e visa a separação de responsabilidades, permitindo uma extensão mais fácil de partes da arquitetura, bem como possibilitando a inclusão de componentes opcionais, quando necessário.

Gerência de contexto

A gerência de contexto é responsável por gerenciar o estado (contexto) para cada um dos clientes. Para isso, obtém a descrição dos contextos (complexos) de interesse da aplicação, registra-se no middleware de provisão de contexto, e atualiza o estados destes clientes quando da recepção de uma notificação de mudança de contexto.

Os contextos de interesse são determinados pela Gerência de Regras, e a Gerência de Contexto utiliza a interface com middleware de provisão de contexto para obter os contextos individuais, interagindo com este middleware de acordo com sua forma de descrição de contextos e sua comunicação. Caso essa comunicação seja do tipo request/reply, o componente deve consultar a situação dos clientes, por exemplo, periodicamente ou no momento que acontece o encaminhamento de uma mensagem para um grupo de clientes. Este tipo de comunicação não é o mais adequado, visto que aumenta o tempo para definição (escolha) das adaptações a serem aplicadas. Uma solução mais adequada é que a gerência de contexto armazene localmente as informações dos contextos correntes de todos os clientes. Mas, neste caso, estas informações devem estar sempre atualizadas. Para isso, a comunicação deve se dar através de eventos que informam o momento da ocorrência de cada situação de contexto, para cada cliente. Também é preciso que seja informado o evento reverso, ou seja, informar quando uma dada situação de contexto deixou de ser válida.

Gerência de regras de adaptação

O componente gerência de regras de adaptação obtém as regras de adaptação (contextos que disparam as adaptações) da camada de aplicação. As regras são interpretadas e são identificadas as situações de contexto que a aplicação deseja tratar, e as adaptações necessárias em cada uma destas si-

tuações. Os contextos de interesse são repassados para a Gerência de Contexto, e as adaptações para a Gerência de Execução de Adaptações.

A principal função deste componente é definir a cadeia de adaptações (transformadores) necessárias para cada cliente a partir das informações de contexto, informações do conteúdo (meta-dados) e das regras. No momento em que há uma nova mensagem para ser enviada aos clientes, a Gerência de Regras consulta a Gerência de Contexto (que avalia o contexto de cada cliente), e então determina a necessidade, ou não, de adaptações de conteúdo para a mensagem. As cadeias de adaptações necessárias para cada cliente servem como entrada para a Gerência de Execução de Adaptações. Essa interação entre os componentes de gerência do serviço de adaptação pode ser vista na Figura 4.3. Esse modelo de cadeia de adaptadores suporta um paradigma de composição de adaptações que permite uma maior flexibilidade na distribuição, na agregação de serviço, enquanto provê fácil reconfiguração em resposta às características variáveis do ambiente sem fio. Para adaptação de um conteúdo com texto e imagens, um exemplo de cadeia de adaptadores, pode ser primeiro a redução da resolução das imagens, e em seguida o rearranjo delas no texto. É interessante também que as regras definam prioridades, ordem e composição de diferentes adaptadores. Esta gerência também é acionada no momento da mudança de um contexto, onde de acordo com as regras pode-se ativar ou desativar adaptações (de funcionamento), por exemplo iniciar armazenamento de informações na ocorrência de uma desconexão.

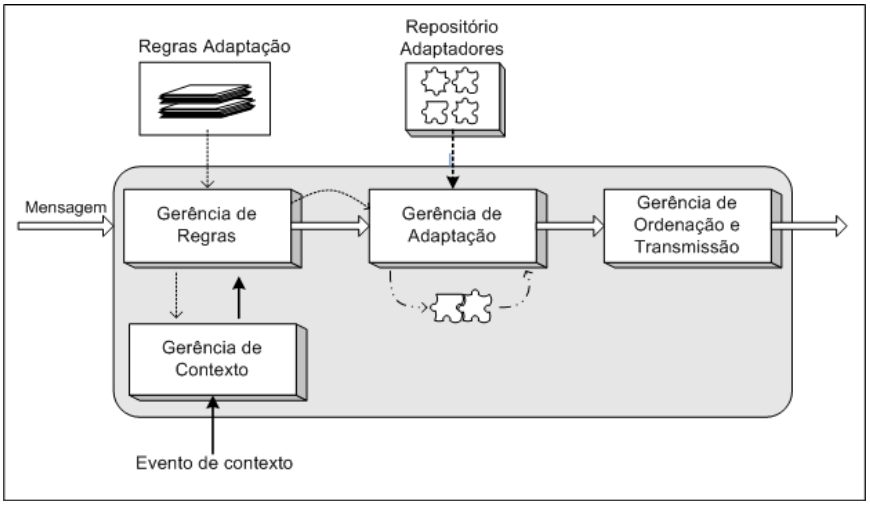

Figura 4.3: Interação entre componentes da arquitetura

A gerência de regras deve prover suporte à atualização das regras de adaptação. Essa atualização é necessária para: (a) melhor adequação ou correção da definição de uma situação de interesse (contexto); (b) para

a criação de novas situações (contextos) de interesse e permitir relacioná-las com as adaptações existentes; ou (c) para atualização ou inclusão de novos módulos de adaptação. A alteração de uma situação de interesse é importante para permitir a inclusão de novos tipos contextos (providos por novos serviços), bem como aprimorar as condições-gatilho de acordo com a evolução da aplicação, ou surgimento de novas necessidades da mesma. Já permitir alteração/substituição dos adaptadores utilizados é importante para o uso de novas técnicas ou algoritmos de transformação, bem como a adição de novos módulos de adaptação (adaptadores), por exemplo, específicos para novos dispositivos ou para um novo tipo de formato de mídia. Dessa forma, prover facilidade de reconfiguração dinâmica do sistema para evolução do mesmo.

Uma funcionalidade desejável é permitir a atualização das políticas de adaptação durante o funcionamento da aplicação, sem a necessidade de pará-la ou reiniciá-la, permitindo assim um funcionamento ininterrupto da aplicação a despeito da inclusão de novos formatos de conteúdos, tipos de dispositivos, etc. Para isso, é necessário que haja mecanismos para uma garantia transacional da execução das adaptações com relação às mensagens sendo processadas. A fim de evitar inconsistências, precisa-se garantir que se uma mensagem já está sendo adaptada/transformada segundo uma cadeia de adaptações quando ocorre uma atualização de regra, a mensagem terminará a seqüência de adaptações iniciada anteriormente.

Gerência de execução adaptações

O componente de gerência de adaptações é responsável pela execução das adaptações. Ele recebe da gerência de regras as cadeias de adaptações que devem ser aplicadas ao conteúdo para atender às necessidades de cada cliente. A partir dessas cadeias, o Gerente de Execução de Adaptadores é responsável por associar as adaptações aos adaptadores mais adequados, carregá-los e executá-los de uma forma eficiente. Caso o adaptador não esteja disponível, ele deve fazer sua carga dinâmica, buscando o adaptador no repositório definido pela camada de aplicação.

Para se definir o caminho de adaptações (ou seja, a seleção dos adaptadores que serão utilizados na cadeia definida pela gerência de regras), pode-se utilizar simplesmente uma tabela de *lookup* para mapear as adaptações aos adaptadores correspondentes [94], ou utilizar as meta-informações dos adaptadores, como tipo (formato) de conteúdo de entrada e saída, verificando quais os adaptadores disponíveis podem ser combinados para se obter o resultado desejado (conteúdo adaptado). Alguns trabalhos formalizam a descrição de

adaptações e adaptadores, e sugerem a criação de grafos com todas as possibilidades de interligação de adaptadores, segundo a cadeia desejada. Após a determinação deste grafo, escolhe-se o melhor caminho segundo um critério de QoS, ou custo, ou tempo de execução [75, 99, 93].

Este é o componente da arquitetura cuja execução envolve a maior complexidade computacional, em relação a tempo de execução, principalmente devido às múltiplas adaptações necessárias para os clientes de uma mesma notificação. Como a customização de conteúdo é específica para cada cliente, de acordo com seu contexto corrente, em princípio seria preciso executar cada cadeia de adaptações individualmente para cada cliente. Entretanto como isso tornaria o sistema muito ineficiente, é necessário utilizar algoritmos para otimização da execução dessas adaptações [4]. Esses algoritmos são detalhados na Seção 4.3.

Gerência de desconexão

A Gerência de Desconexão requisita informações de desconexão da Gerência do Contexto, e pode na ocorrência de desconexões, ativar um armazenamento temporário (*buffering*) de mensagens do cliente desconectado. Deve-se suportar uma política flexível de *buffering*, permitindo que esta seja definida e implementada na camada da aplicação de acordo com suas necessidades, por exemplo, pode-se armazenar somente as n últimas mensagens, ou estas podem ter datas de expiração. A estratégia de fila (buffer) mais simples é eliminar as mensagens para os clientes desconectados. Uma política mais complexa seria armazenar o conteúdo não entregue para tentativas posteriores, permitindo-se definir propriedades como prioridades e tempos de expiração para cada canal.

Além disso, é necessário prover meios para a definição e distinção entre desconexões temporárias ou permanentes, e a adoção de políticas diferenciadas para esses tipos de desconexões. É importante notar que a aplicação pode usar diferentes tipos de informações de contexto para definir 'desconexão', e não apenas qualidade do canal de comunicação sem fio. Por exemplo, uma aplicação pode desejar que as mensagens sejam armazenadas (para uma posterior entrega) sempre que o nível de energia da bateria do dispositivo cliente estiver abaixo de 15%.

[4]Nos casos em que o conteúdo pode ser disseminado mais de uma vez, pode-se adotar na Gerência de Execução o uso de *caches* com conteúdo adaptado, em conjunto com os algoritmos de otimização. Esse cache deve armazenar tanto o conteúdo, quanto a identificação de quais as adaptações sofridas. Ao receber a mensagem, o gerente, antes de ativar as adaptações, verificaria primeiro no cache a existência do conteúdo já adaptado pela mesma sequência de adaptações (ou parte delas).

Gerência de prioridade

A gerência de prioridade é um componente opcional da arquitetura. Sem este elemento, as notificações de conteúdo provenientes do middleware Pub/Sub são tratadas (adaptadas) a medida em que chegam à camada de adaptação. Entretanto, este componente pode ser utilizado para redefinir a política de escalonamento, ou seja, a ordem com que as mensagens serão adaptadas. O critério de prioridade deve ser definido pela aplicação de acordo com seus objetivos e particularidades. Este critério pode ser baseado no conteúdo da mensagem, no cliente destinatário ("cliente VIP") ou em alguma métrica de QoS. Por exemplo, para uma aplicação multimídia com transferência de som, texto e imagem, poder-se-ia definir um critério no qual toda mensagem de som tem precedência a mensagens com texto ou imagem.

Gerência de ordenação e transmissão

A gerência de ordenação e transmissão é responsável pela entrega de mensagens aos clientes utilizando para isso Interface com Middleware Pub/Sub.

A mensagem original, interceptada pela camada de adaptação, provavelmente teve seu conteúdo alterado por adaptações. Além disso, o conjunto de clientes destinatários originais também pode ter sido dividido em subgrupos, para os quais foram aplicados adaptações distintas de conteúdo. Dessa forma, cada mensagem original pode ter várias réplicas modificadas ao final da etapa de adaptação, sendo necessário que este componente informe ao middleware Pub/Sub qual réplica deve ser disseminada a que subconjunto de clientes.

A ordenação de entrega é uma funcionalidade opcional. Esta ordenação visa garantir que as mensagens sejam repassadas ao middleware Pub/Sub na mesma ordem em que chegaram à camada de adaptação. A forma de ordenação deve ser definida de acordo com as necessidades da aplicação. Cada notificação de conteúdo (mensagem) é destinada a clientes que se inscreveram em um tópico/canal. Cada cliente pode ter requisitado notificações de diferentes canais. Há, portanto, diversas possibilidades de ordenação: por cliente, por tópico de notificação, por tipo de conteúdo, ou combinação destes, etc.

O requisito de ordenação de entrega de mensagens impacta em diferentes pontos da arquitetura. Primeiro, na chegada da mensagem à camada de adaptação, deve-se adotar algum tipo de marcação para manutenção da ordem, como *timestamps* ou contadores, dependendo também do tipo de ordenação adotada. Depois, impacta na iteração com a Gerência de Prioridade que escalona mensagens para serem adaptadas, podendo modificar a ordem de entrega. Por fim, antes do envio da mensagem, momento em que pode ser

necessário retardar o envio, esperando mensagens anteriores ficarem prontas e serem encaminhadas.

4.3 Algoritmos de Gerência de Execução de Adaptações

O modelo de comunicação publish/subscribe é do tipo 1-para-muitos, ou seja, uma mesma mensagem é enviada para um conjunto de clientes, i.e, os assinantes de um tópico (subject) publish/suscribe. Porém, em aplicações adaptativas sensíveis a contexto, a personalização de conteúdo das mensagens deve ser realizada de acordo com o contexto de cada cliente. Portanto, o desafio é permitir esse tipo de personalização em uma comunicação publish/subscribe.

Como mencionado anteriormente, para a entrega customizada de conteúdo, é necessário utilizar adaptações específicas para cada cliente de acordo com seu estado de contexto. A solução mais simples seria quebrar a comunicação multiponto para vários envios ponto a ponto, replicando a mensagem para cada cliente e então aplicar as adaptações. Essa solução força bruta é dispendiosa e ineficiente e, no caso de adaptações que demandam muitos recursos computacionais, pode tornar o sistema não escalável.

Pode-se perceber, então, que na arquitetura proposta o componente mais custoso (tempo e espaço) é a Gerência de Execução de Adaptações, responsável pela execução das adaptações adequadas aos clientes. Esta seção discute algoritmos que têm por objetivo otimizar (reduzir) o custo de adaptação, reduzindo também o número de mensagens replicadas para permitir uma maior escalabilidade do sistema. A idéia é realizar o maior número possível de adaptações comuns, reduzindo a execução repetida de adaptadores. A solução proposta é dividir o grupo original em subgrupos de clientes com características comuns de acordo com suas condições atuais de contexto, nos quais o mesmo conjunto de adaptadores é utilizado.

A seguir são formalizados o problema e a solução proposta. Inicialmente, é apresentado um algoritmo básico para adaptação de conteúdo para um único cliente, e, posteriormente, são apresentados dois algoritmos para gerenciar as adaptações de forma correta e eficiente [118]. Um desses algoritmos é utilizado na implementação do componente Gerência de Execução de Adaptações, discutido na Seção 4.2. Serão feitas também análises de complexidade do pior caso.

4.3.1 Resumo das Definições

Algumas definições de termos utilizados na descrição dos algoritmos são resumidas a seguir:

Atributo de contexto representa uma característica do ambiente, do dispositivo, ou da rede. Alguns exemplos de atributos de contexto são: tipo de dispositivo, grau de utilização da CPU, qualidade da conexão, tamanho da tela.

Condição de Contexto, ou também chamada situação de contexto, envolve a comparação de atributos de contexto com valores constantes, e também a combinação desses atributos definida segundo uma expressão lógica.Ex: "`Tipo de dispositivo = PDA & utilização da CPU` $>$ `60%` ".

Estado da condição de contexto do cliente indica a situação de um cliente em relação a uma dada condição de contexto num determinado momento, indicando estado ativo, i.e a expressão de contexto é verdadeira, ou estado inativo.

Adaptação de conteúdo é a transformação realizada sobre o conteúdo de uma mensagem através de adaptadores. As adaptações são acionadas de acordo com as condições de contexto do cliente ativas no momento. Define-se dois tipos de adaptações:

1. *Parametrizadas por atributos de contexto*: utilizam o valor corrente de atributos de contexto como parâmetros para adaptação. Por exemplo, o atributo de contexto localização corrente pode ser usado para se colocar uma indicação em um mapa, ou filtrar parte da informação, ou ainda dar um zoom, ou centralizar o mapa na posição atual. Em outro exemplo, o tamanho da tela pode ser usado para redimensionar uma imagem para o tamanho da tela do dispositivo.

2. *Não parametrizadas*: ativados por condições de contexto, mas não utilizam o valor corrente de atributos de contexto para efetuar a adaptação. Esses adaptadores em geral têm parâmetros pré-definidos (ou configurados nas regras), por exemplo, um adaptador para reduzir a qualidade da imagem em 50% se a qualidade da rede estiver baixa, ou converter imagem para preto e branco. Outro exemplo seria adaptador que realize uma sumarização de texto dado a condição que o tipo de dispositivo é um celular.

4.3.2 Adaptação de conteúdo baseado em contexto de clientes móveis

O sistema proposto adota a abordagem de adaptação dinâmica de conteúdo. Como já mencionado, nesta abordagem a adaptação é executada apenas no momento do envio do conteúdo para o cliente, de acordo com o contexto corrente do cliente.

Para execução das adaptações de conteúdo é necessário que sejam definidas, além das próprias adaptações, as condições de contexto de interesse que acionam tais adaptações.

Denota-se a seqüência das n situações (ou condições) de contexto de interesse de:

$$S = (s_1, s_2, \ldots, s_n) \tag{4-1}$$

O elemento responsável por executar uma adaptação de conteúdo é chamado adaptador, e o conjunto de adaptadores disponíveis de $A = a_1, a_2, \ldots, a_d$.

Para cada condição de contexto, existem alguns adaptadores de A que são ativados por esta condição em uma determinada ordem. Denota-se $s_i \longrightarrow a$, a situação de contexto s_i que dispara a adaptação a. A seqüência de adaptadores ativados por s_i, A_{Si} [5], é definida por:

$$A_{Si} = (a \in A \,/\, s_i \longrightarrow a), \quad s_i \in S \tag{4-2}$$

O estado de todas as condições de contexto do cliente no momento da chegada da mensagem ao sistema é dado pela tupla E_c:

$$E_c = < e_{s_1}, e_{s_2}, \ldots, e_{s_n} >, \quad e_{s_i} \in \{\text{ativo}\,,\, \text{inativo}\} \tag{4-3}$$

onde e_{s_i} indica o estado da situação de contexto s_i com relação aos atributos de contexto do cliente em determinado momento.

A tupla E_c do cliente é mantida pelo módulo Gerenciador de Contexto, que é responsável por avaliar todas as situações de contexto S para o cliente, a partir da interação com um serviço de contexto. Conforme os eventos de atualização de contexto são recebidos este gerenciador atualiza as informações de E_c.

Considera-se que todas as adaptações de um conjunto A_{Si} são executadas quando o estado correspondente s_i do cliente estiver ativo, ou seja, e_{s_i} = ativo. Além disso, existe uma ordem de aplicação das seqüências A_{Si} que é determinada pela ordem das condições de contexto na seqüência S.

Denota-se A_c a seqüência de adaptadores a serem aplicados a uma mensagem transmitida ao cliente c em um determinado instante. Esta seqüência

[5] Assume-se que a sequência de adaptadores não é comutativa.

é definida de acordo com os estados das situações de contexto do cliente e, portanto, deve ser calculada a cada momento em que uma nova mensagem chega ao sistema para ser enviada para o cliente. Assim, A_c é a concatenação ($\oplus$) das seqüências A_{Si} cuja situação s_i está ativa para o cliente no instante do envio da mensagem:

$$A_c = A_{S_{i_1}} \oplus A_{S_{i_2}} \oplus \ \ldots \ \oplus A_{S_{i_l}} \tag{4-4}$$

onde $\{s_{i_1}, \ldots, s_{i_l}\}$ é o subconjunto de S formado pelas condições de contexto de interesse s_{i_j} tais que s_{i_j} = ativo.

Desta forma, antes que a mensagem seja encaminhada ao cliente, deve-se efetuar todas as adaptações definidas na seqüência de adaptadores A_c. Só ao término de todas as adaptações, a mensagem será enviada ao cliente. A ação de adaptação de conteúdo baseado em contexto para um cliente c pode ser descrita pelos seguintes passos:

1. Verificar o estado de todas as condições de contexto do cliente (E_c);
2. Identificar as situações de contexto ativas no momento e criar a seqüência de adaptadores a serem utilizados (A_c) como na definição 4-4;
3. Efetuar as adaptações $a \in A_c$

O Algoritmo 1 apresenta a operação do sistema ao receber uma mensagem a ser encaminhada ao cliente. Neste momento, o algoritmo é invocado com os seguintes argumentos: a mensagem m, o cliente c a que a mensagem se destina e a primeira condição de contexto a ser analisada (S_1), ou seja, $i = 1$. Assim, a chamada inicial do algoritmo é:

$$AdapterMngr(c, m, 1)$$

Considera-se que os conjuntos S, A_{Si}, E_c são conhecidos globalmente, e foram previamente definidos e configurados. Além disso, a ordenação dos adaptadores também já foi efetuada.

Algoritmo 1: Adaptação de conteúdo baseada no contexto de um cliente

Entrada:

c - cliente

msg - mensagem a ser enviada

i - índice da condição de contexto analisada, $i = 1 \ldots |S|$

Resultado: mensagem adaptada se necessário e entregue ao cliente

```
AdapterMngr (c, msg, i)
  if i > |S| then
    Send(msg, c)     /* Envia mensagem adaptada ao cliente */
    return
  end
  if e_{s_i} = ativo then
    /* Aplicar adaptações referentes a situação de contexto
       S_i                                                   */
    msg' ⟵ ExecuteAdapters(A_{Si}, msg)
    /* Executar o algoritmo para próxima condição de
       contexto, sobre a mensagem adaptada                   */
    AdapterMngr(c, msg', i + 1)
  else
    /* Executar o algoritmo para próxima condição de
       contexto                                              */
    AdapterMngr(c, msg, i + 1)
  end
end
```


Complexidade do algoritmo

O algoritmo 1 é recursivo sobre a seqüência de situações de contexto S. São utilizados dois métodos auxiliares: (i) `Send`, para a entrega da mensagem ao cliente e (ii) `ExecuteAdapters` que é responsável por executar a seqüência de adaptações sobre uma dada mensagem.

A complexidade do Algoritmo1 em relação ao número de adaptações efetuadas é analisada a seguir. Denomina-se o número de condições de contexto de $n = |S|$ e o número total de adaptadores de $d = \sum |A_{Si}|$, $i = 1 \ldots n$. Como a execução dos adaptadores depende do estado da condição de contexto do cliente, define-se:

$$f(S_i) = \begin{cases} 0, & \text{se } e_{s_i} = \text{inativo} \\ |A_{Si}|, & \text{se } e_{s_i} = \text{ativo} \end{cases}$$

Como em qualquer algoritmo recursivo, o algoritmo anterior gera uma relação de recorrência, de modo que a complexidade de qualquer rotina é calculada como a complexidade dentro da própria rotina, acrescido da complexidade das chamadas recursivas [119, 120].

Assim, a complexidade é calculada em função do número de adaptações, através da equação de recorrência:

$$\begin{aligned} T(n) = &\ f(S_n) + T(n-1) \\ = &\ f(S_n) + [f(S_{n-1}) + T(n-2)] \\ &\ \vdots \\ = &\ f(S_n) + f(S_{n-1}) + \ldots + f(S_1) + T(0) \end{aligned}$$

$\therefore$

$$T(n) = \sum_{i=1}^{n} f(S_i)$$

Como a complexidade, em relação ao número de adaptadores executados, pode variar de 0 a $\sum_1^n |A_{Si}|$, dependendo do número de condições de contexto do cliente com estado ativo no momento da invocação do algoritmo, tem-se que:

$$T(n) = O(d) \tag{4-5}$$

Este tempo deve ser multiplicado pelo custo das adaptações.

4.3.3 Algoritmo para adaptação de conteúdo em sistemas Publish/Subscribe

A seção anterior apresentou um algoritmo básico para adaptação de conteúdo baseado no contexto corrente do cliente. Entretanto, para comunicação Pub/Sub uma mesma mensagem é disseminada a um conjunto de clientes, e a entrega personalizada deve seguir as condições específicas de cada um destes clientes. Uma solução simples é quebrar a comunicação multiponto em vários envios ponto a ponto, replicando a mensagem para cada cliente, e então aplicar as adaptações. Ou seja, o Algoritmo 1 seria aplicado para cada cliente e então ter-se-ía um custo de $O(\text{nº de clientes} \times d)$. Mas esta solução é dispendiosa e pouco eficiente.

Nesta seção, é apresentado um algoritmo, chamado *Context-Aware Client Grouping Algorithm*, para otimizar o custo de adaptação das mensagens em sistemas Pub/Sub. O algoritmo reduz o número de adaptações executadas, bem como o número de mensagens replicadas. Para isso, é necessário identificar clientes com condições de contexto semelhantes, e então agrupá-los de forma a executar o menor número possível de adaptações repetidas.

O algoritmo considera que uma mensagem deve ser encaminhada a um grupo de clientes. Este grupo é determinado pelo sistema Pub/Sub, de acordo com o tópico (e condições) de interesse nos quais os clientes se inscreveram. De acordo com as condições de contexto comuns devem ser criados grupos de clientes para os quais a mensagem será adaptada.

$C = \{c_1, c_2, \ldots, c_K\}$ é o conjunto de clientes para o qual a mensagem está endereçada.

Como anteriormente, S indica a seqüência de condições de contexto e A_{Si} representa a seqüência de adaptadores ativados pela condição s_i.

Aqui o estado das condições de contexto de todos os clientes de C no momento da chegada da mensagem ao sistema é dado pelo conjunto E_C. Este conjunto é formado pelas tuplas E_{c_k} que representam o estado de cada cliente c_k. Assim,

$$E_{c_k} =< e_{c_k,s_1}, e_{c_k,s_2}, \ldots, e_{c_k,s_n} >, \quad e_{c_k,s_i} \in \{\text{ativo}\,,\, \text{inativo}\}$$
$$E_C = \bigcup_{c_k \in C} E_{c_k} \tag{4-6}$$

onde e_{c_k,s_i} indica o estado da situação de contexto s_i com relação aos atributos de contexto do cliente c_k em determinado momento.

O Gerenciador de Contexto (Seção 4.2) é responsável por avaliar todas as situações de contexto S para cada cliente e atualizar as informações de E_C, conforme os eventos de atualização de contexto são recebidos.

Similarmente à equação 4-4, denota-se por A_{c_k} a seqüência de adaptadores a serem aplicados a uma mensagem transmitida ao cliente c_k em um determinado instante. O conjunto de destas sequências para todos os clientes é chamado de A_C. Assim:

$$A_{c_k} = A_{S_{i_{1_k}}} \oplus A_{S_{i_{2_k}}} \oplus \ \ldots \ \oplus A_{S_{i_{l_k}}}$$
$$A_C = \bigcup A_{c_k},\ \forall\, c_k \in C \tag{4-7}$$

onde $\{s_{i_{1_k}}, \ldots, s_{i_{l_k}}\}$ é o subconjunto de S formado pelas condições de contexto de interesse $s_{i_{j_k}}$ tais que s_{i_j} = ativo para o cliente c_k.

O sistema deve enviar a mensagem adaptada a cada cliente, mas efetuando o menor número possível de adaptações repetidas. Para isso, é necessário identificar os adaptadores entre as seqüências A_{c_k} que são comuns, e que ocorrem na mesma ordem. Estas adaptações comuns são chamadas de *prefixos* Pr, e os clientes com mesmo prefixo são separados em grupos, de tal forma que estas adaptações comuns sejam realizadas apenas uma única vez para cada

grupo.

Desta forma, a Gerência de Adaptação efetua o processo descrito a seguir, para um grupo G_C de clientes:

1. Verificar o estado de todas as condições de contexto (E_{c_k}) de cada cliente c_k em G_C;

2. Identificar as situações de contexto ativas no momento e criar a seqüência de adaptadores a serem utilizados (A_{c_k}) para cada cliente c_k em G_C;

3. Verificar os adaptadores comuns entre os clientes, dividindo-os em grupos:

   (a) Identificar os *prefixos* (Pr) comuns entre todos os A_{c_k}. Dois clientes c_i e c_j estarão em um mesmo grupo se possuírem prefixo comum [6].

4. Efetuar as adaptações em cada grupo:

   (a) Replicar a mensagem original para cada grupo, e executar as adaptações correspondentes.

Os passos acima são descritos no Algoritmo 2, no qual a iteração ocorre em relação ao estado de cada condição de contexto dos clientes.

Os grupos de clientes são denominados $G_C \in P(C)$, onde P(C) é o conjunto de todos os possíveis subconjuntos de C. Define-se G_{Si} como o grupo de clientes de G_C que estão com a situação de contexto s_i ativa, e $G_{\neg Si}$ os clientes com condição s_i inativa.

Inicialmente, os argumentos de invocação do algoritmo são o conjunto total de clientes a que a mensagem m se destina, i.e, $G_C = C$; e a primeira condição de contexto S_1, ou seja, $i = 1$. Assim, a chamada inicial do algoritmo é:

$$AdapterMngr(C, m, 1)$$

[6] Gera uma árvore de prefixos.

Algoritmo 2: Adaptação de conteúdo para um grupo de clientes (*Context-Aware Client Grouping Algorithm*)

Entrada:

G_C - grupo de clientes para o qual a mensagem deve ser enviada

msg - mensagem a ser enviada, e adaptada se necessário

i - índice da condição de contexto analisada, $i = 1 \ldots |S|$

Resultado: mensagem adaptada e entregue aos clientes

AdapterMngr (G_C, msg, i)

if $i > |S|$ **then**

Send(msg, G_C) `/* Envia mensagem adaptada ao grupo de clientes` G_C `*/`

return

end

`/* Identificar quais clientes de` G_C `estão com a situação de contexto` i `ativa e dividí-los em dois grupos: (i) clientes com situação` i `ativa e (ii) clientes com condição não ativa */`

$G_{Si} \longleftarrow \{c_k : c_k \in G_C \ / \ e_{c_k,S_i} = \text{ativo}\}$

$G_{\neg Si} \longleftarrow \{c_k : c_k \in G_C \ / \ e_{c_k,S_i} = \text{inativo}\}$

if $G_{Si} \neq \oslash$ **then**

`/* Aplicar adaptações referentes à situação de contexto` S_i `*/`

$msg' \longleftarrow$ ExecuteAdapters(A_{Si}, msg)

`/* Executar o algoritmo, sobre a mensagem adaptada, para próxima situação de contexto */`

AdapterMngr$(G_{Si}, msg', i + 1)$

end

if $G_{\neg Si} \neq \oslash$ **then**

`/* Executar o algoritmo para próxima situação de contexto */`

AdapterMngr$(G_{\neg Si}, msg, i + 1)$

end

end

Uma observação é que no Algoritmo 2 o método `Send` é responsável por enviar a mensagem ao grupo de clientes especificado.

Exemplo de execução

O exemplo a seguir ilustra a execução do algoritmo descrito acima. Neste exemplo, uma mensagem deve ser enviada a um conjunto de clientes indicado pelo gerenciador Pub/Sub. Para simplificar o exemplo, considera-se que cada situação de contexto torna ativo apenas um adaptador. Os parâmetros considerados são:

- Clientes: $C = \{c_1, c_2, ..., c_6\}$
- Situações de contexto: $S = (s_1, s_2, ..., s_5)$.
- Adaptadores: $A = \{A_1, A_2, ..., A_5\}$
- Adaptadores ativados por cada situação de contexto:
  - $A_{S_1} = (A_1)$
  - $A_{S_2} = (A_2)$
  - $A_{S_3} = (A_3)$
  - $A_{S_4} = (A_4)$
  - $A_{S_5} = (A_5)$
- Estado da situação de contexto de cada cliente no momento do envio da mensagem. Lembrando que a tupla $E_{c_k} =< e_{c_k,s_1}, e_{c_k,s_2}, \ldots, e_{c_k,s_5} >$ indica o estado de cada condição de contexto para o cliente k.
  - E_{c_1} = <ativo, ativo, inativo, ativo, ativo>
  - E_{c_2} = <inativo, ativo, inativo, ativo, inativo>
  - E_{c_3} = <inativo, inativo, ativo, inativo, inativo>
  - E_{c_4} = <ativo, ativo, inativo, ativo, inativo>
  - E_{c_5} = <ativo, inativo, inativo, inativo, ativo>
  - E_{c_6} = <inativo, ativo, inativo, ativo, inativo>

Por exemplo, para uma aplicação que transmitisse as transparências (composta de textos e imagens) de uma palestra, as situações de interesse poderiam ser: conexão ruim (s1), em movimento rápido (s2), energia inferior a 10% (s3), luminosidade elevada (s4) e incompatibilidade do formato transmitido (s5). Para essas situações os seguintes adaptadores poderiam ser acionados: redução da qualidade de imagens em 60%, aumento da imagem e fonte do texto, remoção de imagens, aumento do contraste, conversão de formatos (por exemplo, jpeg para tiff), respectivamente.

A Tabela 4.1 ilustra o exemplo especificado acima, indicando a seqüência de adaptações que deve ser aplicada à mensagem a ser entregue a cada cliente. Cada célula da tabela equivale a um estado de contexto e_{c_k,s_i} e o caractere 'x' representa que o estado correspondente está ativo. Assim, cada linha da

tabela apresenta o estado de contexto atual E_{c_k} e, portanto, a seqüência de adaptadores ativados (A_{c_k}) para cada cliente c_k.

	S_1	S_2	S_3	S_4	S_5
	A_1	A_2	A_3	A_4	A_5
E_{c_1}	x	x		x	x
E_{c_2}		x		x	
E_{c_3}			x		
E_{c_4}	x	x		x	
E_{c_5}	x				x
E_{c_6}		x		x	

Tabela 4.1: Seqüências A_{C_k} para o exemplo

Note que seria executado um total de 14 adaptações e 6 mensagens distintas seriam criadas, caso a mensagem fosse apenas replicada para cada cliente.

A seguir são ilustradas a execução do algoritmo e a divisão dos clientes em subgrupos. A Figura 4.4 ilustra a configuração dos grupos após a avaliação da condição de contexto 1. Na Figura 4.5 é apresentada a segunda iteração do algoritmo. A execução continua até que todas as situações de contexto tenham sido avaliadas e as respectivas adaptações aplicadas, quando necessário.

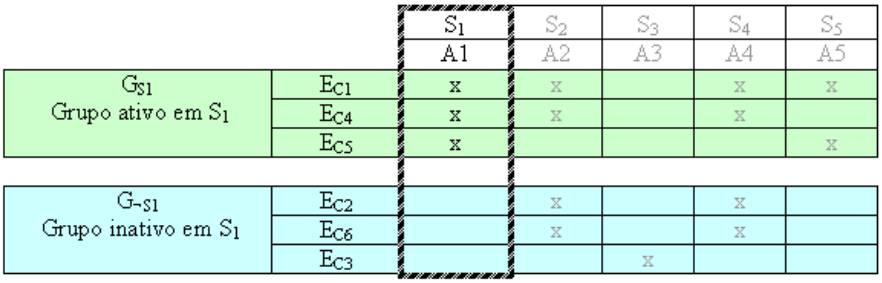

Figura 4.4: Passo 1 do algoritmo

Para melhor demonstrar a formação de grupos, a replicação da mensagem apenas nos momentos realmente necessários e a aplicação dos adaptadores, a Figura 4.6 apresenta a árvore de divisão de grupos e fluxo da adaptação da mensagem para o exemplo anterior.

		S_1	S_2	S_3	S_4	S_5
		A1	A2	A3	A4	A5
G_{S1} ativo em S_2	E_{C1}	x	x		x	x
	E_{C4}	x	x		x	
G_{S1} inativo em S_2	E_{C5}	x				x
$G_{\neg S1}$ ativo em S_2	E_{C2}		x		x	
	E_{C6}		x		x	
$G_{\neg S1}$ inativo em S_2	E_{C3}			x		

Figura 4.5: Passo 2 do algoritmo

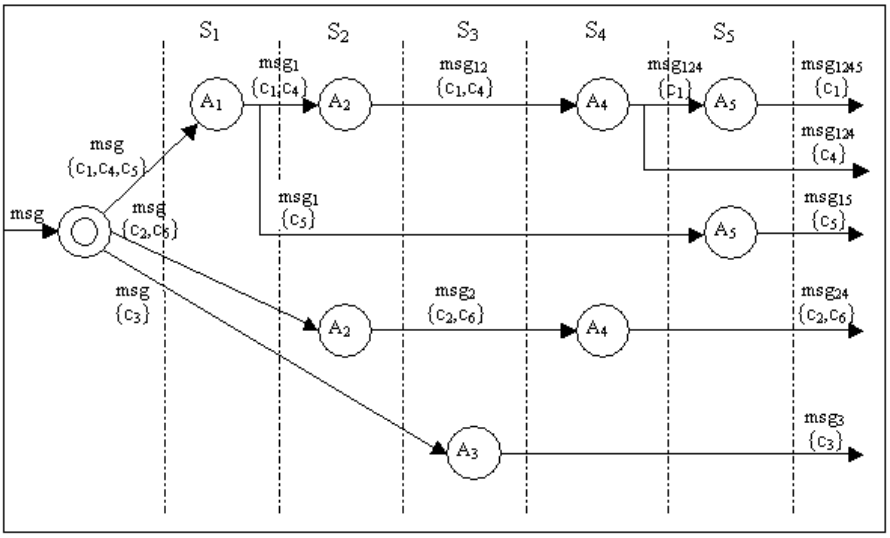

Figura 4.6: Envio de mensagem a um grupo Pub/Sub

Com a utilização do algoritmo proposto (Algoritmo 2), pode-se perceber que o número de adaptações é reduzido de 14 para 8 (redução de aproximadamente 40%) e são criadas 5 mensagens distintas.

Avaliação do algoritmo

Como dito anteriormente, o Algoritmo 2 tem como objetivo reduzir o número total de adaptações realizadas, que seria, no caso do algoritmo mais simples, da ordem do número de clientes multiplicado pelo número de adaptadores. Esta otimização (ou % de melhoria) do número de adaptações executadas depende de vários fatores, sendo o principal deles o número de clientes com contextos similares, bem como a ordem de execução dos adaptações. Ou seja, é dependente do número de clientes com prefixos comuns de adaptadores, e a quantidade de adaptações relativas a esses prefixos.

Denomina-se o número de clientes $k = |C|$, o número de condições de contexto $n = |S|$, e o número total de adaptadores de $d = |A| = \sum |A_{Si}|$, $i = 1 \ldots n$. Como a execução dos adaptadores depende do estado da condição de contexto dos clientes do grupo, define-se:

$$f(S_i) = \begin{cases} 0, & \text{se } |G_{Si}| = 0 \\ |A_{Si}|, & \text{se } |G_{Si}| > 0 \end{cases}$$

Para calcular a complexidade do Algoritmo 2 em relação ao número de adaptações realizadas, define-se g como o fator de divisão de grupos, ou seja, a porcentagem de clientes com situação s_i ativa. A complexidade é dada pela seguinte equação de recorrência, em função do número de situações de contexto e número de clientes em cada grupo:

$$\begin{aligned}
T(n,k) &= f(S_1) + T(n-1, \tfrac{k}{g}) + T(n-1, k - \tfrac{k}{g}) \\
&= f(S_1) + T(n-1, \tfrac{k}{g}) + T(n-1, \tfrac{k(g-1)}{g}) \\
&= f(S_1) + [f(S_2) + T(n-2, \tfrac{k}{g} \cdot \tfrac{1}{g}) + T(n-2, \tfrac{k}{g} - \tfrac{k}{g} \cdot \tfrac{1}{g})] + \\
&\quad [f(S_2) + T(n-2, \tfrac{k(g-1)}{g} \cdot \tfrac{1}{g}) + T(n-2, \tfrac{k(g-1)}{g} - \tfrac{k(g-1)}{g} \cdot \tfrac{1}{g})] \\
&= f(S_1) + 2f(S_2) + T(n-2, \tfrac{k}{g^2}) + 2T(n-2, \tfrac{k(g-1)}{g^2}) + \\
&\quad T(n-2, \tfrac{k(g-1)^2}{g^2}) \\
&= f(S_1) + 2f(S_2) + 4f(S_3) + \\
&\quad [T(n-3, \tfrac{k}{g^3}) + 3T(n-3, \tfrac{k(g-1)}{g^3}) + 3T(n-3, \tfrac{k(g-1)^2}{g^3}) + \\
&\quad T(n-3, \tfrac{k(g-1)^3}{g^3})] \\
&\vdots \\
&= f(S_1) + 2f(S_2) + 4f(S_3) + \ldots + 2^{i-1} f(S_i) + \\
&\quad \textstyle\sum_{j=0}^{i} c_j \, T(n-i, \tfrac{k(g-1)^j}{g^i}) \\
&\vdots \\
&= \textstyle\sum_{i=1}^{n} 2^{i-1} f(S_i) + \sum_{j=0}^{n} c_j \, T(0, \tfrac{k(g-1)^j}{g^n})
\end{aligned}$$

Na equação acima, os coeficientes c_j são os coeficientes da série $(x+1)^i = c_0 + c_1 x + c_2 x^2 + \ldots + c_i x^i$.

Sabendo que $T(0,k) = 0, \ \forall\, k$ e $T(n,0) = 0, \ \forall\, n$, note que complexidade em relação ao número de adaptações realizadas é dada pela equação

$$T(n,k) = \sum_{i=1}^{n} 2^{i-1} f(S_i)$$

Isso apenas é verdadeiro se todos os grupos de clientes da expressão de recorrência tiverem pelo menos um cliente, ou seja, $\frac{k(g-1)^j}{g^n} \geq 1 \, \forall j = 0, \ldots, n$. Caso contrário, a árvore de execução é podada, devido à recursão parar sempre

que um grupo G_{Si} ou $G_{\neg Si}$ é vazio. Desta forma, o número de adaptações na maioria dos casos é menor que o indicado na equação $T(n,k)$ acima.

Deseja-se mostrar que o algoritmo proposto nesta seção é melhor que o algoritmo força bruta, que aplica o Algoritmo 1 para cada um dos k clientes. Na realidade será mostrado que o Algoritmo 2 possui complexidade entre $O(d)$, no melhor caso (admitindo que sempre há adaptações a realizar), e $O(d \times k)$ no pior.

No melhor caso, cada adaptação necessária é executada uma única vez, e apenas uma mensagem é gerada para todo o conjunto de clientes. Isso ocorre quando todos os clientes possuem o mesmo conjunto de situações de contexto ativas, formando sempre, e apenas 1 grupo G_{Si}. Assim a equação de recorrência do melhor caso é dada por:

$$\begin{aligned} T(n,k) &= f(S_1) + T(n-1,k) + T(n-1,0) \\ &= f(S_1) + f(S_2) + T(n-2,k) \\ &\vdots \\ &= \textstyle\sum_{i=1}^{n} f(S_i) \\ &\leq \textstyle\sum_{i=1}^{n} |A_{S_i}| \\ \therefore & \\ T(n,k) &= O(d) \end{aligned}$$

Note que no melhor caso a complexidade do Algoritmo 2 se iguala à complexidade do Algoritmo 1 (equação 4-5), que realiza as adaptações para apenas um cliente.

O Algoritmo 2 efetua a recursão sobre o número de situações de contexto, mas possui duas condições de parada para a recursão: uma é o próprio número de situações e a outra é o número de clientes em um grupo. A primeira condição é sempre fixa ($n = |S|$), mas a segunda condição sempre poda um ramo da execução quando o tamanho do grupo de clientes for zero. Assim, o pior caso para o algoritmo ocorre quando todos os ramos da recursão são executados. Isso ocorre sempre que o grupo de clientes é repartido ao meio, ou seja, quando $g = 2$. Neste caso, a equação de recorrência pode ser escrita como:

$$\begin{aligned}
T(n,k) &= f(S_1) + T(n-1,k/2) + T(n-1,k/2) \\
&= f(S_1) + 2T(n-1,k/2) \\
&= f(S_1) + 2[f(S_2) + T(n-2,k/4) + T(n-2,k/4)] \\
&= f(S_1) + 2f(S_2) + 4T(n-2,k/4) \\
&= f(S_1) + 2f(S_2) + 4[f(S_3) + T(n-3,k/8) + T(n-3,k/8)] \\
&= f(S_1) + 2f(S_2) + 4f(S_3) + 8T(n-3,k/8) \\
&\vdots \\
&= \textstyle\sum_1^i 2^{i-1} f(S_i) + 2^i T(n-i,k/2^i) \\
&\vdots \\
&= \textstyle\sum_{i=1}^n 2^{i-1} f(S_i) + 2^n T(0,k/2^n) \\
&= \textstyle\sum_{i=1}^n 2^{i-1} f(S_i) + 0
\end{aligned}$$

Como $f(S_i) = 0$ ou $f(S_i) = |A_{Si}|$, então $f(S_i) \leq |A_{Si}|$ e assim:

$$T(n,k) \leq \sum_{i=1}^{n} 2^{i-1} |A_{Si}| \tag{4-8}$$

Sabe-se que $\sum_{i=1}^n |A_{Si}| = d$ e que $|A_{Si}| \geq 1 \ \forall s_i \in S$, portanto $|A_{Si}| < d$. Então:

$$\begin{aligned}
T(n,k) &\leq \textstyle\sum_{i=1}^n 2^{i-1} |A_{Si}| \\
&< \textstyle\sum_{i=1}^n 2^{i-1}\, d \\
&= d \textstyle\sum_{i=1}^n 2^{i-1} \\
&= d\, 2^n \\
&\ddots \\
T(n,k) &< d \times 2^n
\end{aligned}$$

Cada grupo de clientes na execução possui pelo menos um cliente, ou seja, $\frac{k(g-1)^n}{g^n} \geq 1$ e como $g = 2$, então $k/2^n \geq 1$, portanto $2^n \leq k$. Assim, para o pior caso, a complexidade é dada por:

$$T(n,k) < d \times k \tag{4-9}$$

Outro pior caso poderia ocorrer se todos os clientes estivessem com contextos totalmente distintos, e então, seria necessário replicar a mensagem para cada cliente, não sendo possível executar (aproveitar) nenhuma adaptação em comum. Entretanto isto só é possível se o número de clientes k for menor que o número de situações de contexto n. Mas neste caso a complexidade do algoritmo é exatamente igual ao do algoritmo para adaptação de um único cliente multiplicado pelo número de clientes: $T(n,k) = O(d \times k)$.

Para delimitar melhor a complexidade do pior caso, considere uma situação bastante comum, onde toda condição de contexto implica em apenas uma adaptação, ou seja $|A_{Si}| = 1 \ \forall s_i \in S$. Desta forma, $d = \sum_{i=1}^n |A_{Si}| = n$.

A complexidade do o algoritmo 1 de adaptação para um cliente, dada pela equação 4-5, passa a ser $O(n)$, e quando aplicada para k clientes, tem-se uma complexidade da ordem de $O(n \times k)$.

Aplicando esta mesma suposição à condição de pior caso, a equação 4-8 torna-se:

$$\begin{aligned} T(n,k) &\leq \sum_{i=1}^{n} 2^{i-1}|A_{Si}| \\ &= \sum_{i=1}^{n} 2^{i-1} \cdot 1 \\ &= \sum_{i=1}^{n} 2^{i-1} \\ &= 2^n - 1 \\ &\therefore \\ T(n,k) &\leq 2^n \end{aligned}$$

Se o número de clientes for maior que o número de grupos, i.e, $k/2^n \geq 1$, então $k \geq 2^n$. Assim, para o pior caso tem-se:

$$T(n,k) \leq k \tag{4-10}$$

que é significativamente melhor que o tempo do algoritmo força bruta, $O(n \times k)$.

Mas se $k < 2^n$, então o número de clientes não é suficiente para completar a árvore de execução, e alguns ramos seriam podados. Entretanto, para o pior caso, considera-se que quando um grupo possui apenas 1 cliente, deste momento em diante este cliente efetuará todas as adaptações das condições de contexto seguintes, ou seja a divisão do grupo para as iterações seguintes será $|G_{Si}| = 1$ e $|G_{\neg Si}| = 0$. Desta forma, nas últimas iterações, os subgrupos terão apenas 1 cliente, e para cada um as adaptações são executadas individualmente em cada estado seguinte. Neste caso as últimas iterações contribuem com peso k na complexidade do algoritmo. Assim, a complexidade passa a ser composta por uma parcela em que as adaptações são aplicadas a grupos de tamanho superior a um, e outra parcela na qual as adaptações são executadas para todos os clientes. Portanto:

$$T(n,k) \leq \sum_{i=1}^{\log k} 2^{i-1}|A_{Si}| + \sum_{i=\log k+1}^{n} k|A_{Si}|$$

lembrando que $\forall s_i \ |A_{Si}| = 1$ e $k < 2^n$, ou seja, $n > \log k$,

$$\begin{aligned} T(n,k) &\leq 2^{\log k} + k(n - \log k) \\ &= k + k(n - \log k) \\ &= k(n - \log k + 1) \qquad (4\text{-}11) \\ &\therefore \\ T(n,k) &< kn \end{aligned}$$

Portanto, a complexidade do algoritmo no pior caso é da ordem de $O(n \times k)$. Mas na prática, a complexidade é menor que $O(n \times k)$, tendendo a $O(k)$, conforme o número de clientes aumenta.

Na maioria dos casos, como será visto no Capítulo 6, há uma melhora significativa. Isto porque é provável que muitas situações de contexto sejam comuns a vários clientes. Por exemplo, para o contexto "tipo de dispositivo = celular" pode-se associar as adaptações "sumarize texto" e "remova imagens". Neste caso é muito provável que haja um grande número de clientes que estão utilizando um aparelho celular, e então as duas adaptações podem ser executadas uma única vez para este subgrupo de clientes. Além disso, em geral o número de clientes tende a ser muito superior ao número de situações de contexto (i.e. $k >> n$); o primeiro podendo chegar a centenas, enquanto que o segundo provavelmente consistirá de uma dezena ou um pouco mais. Portanto, nestes casos sempre haverá grupos de clientes com as mesmas situações, ou seja, clientes com necessidades comuns de adaptações, e que podem ser executadas apenas uma única vez para todo o grupo, ou seja, o caso mais comum é que $T(n,k) << d \times k$.

4.3.4 Algoritmo para otimização de adaptações parametrizadas por contexto

O algoritmo descrito na Seção 4.3.3 considera adaptações não parametrizadas, mas não considera a existência de adaptadores parametrizados por contexto. Os adaptadores parametrizados por contexto não só são acionados por condições de contexto, mas também utilizam os valores correntes de atributos de contexto como parâmetros para a execução da adaptação. O exemplo mais comum de adaptador parametrizado é aquele que utiliza o atributo "tamanho de tela" do dispositivo para efetuar o dimensionamento de uma imagem.

O problema para adaptação de contexto para um grupo de clientes quando se utiliza adaptadores parametrizados é definida com mais detalhes a seguir.

Para cada seqüência A_{S_i}, que indica os adaptadores ativados pela condição de contexto s_i, é preciso identificar quais destes adaptadores são parametrizados, e quais os atributos de contexto utilizados em cada adaptador desse tipo.

Denota-se o tipo do adaptador de $t_a \in \{$parametrizado , não parametri-

zado}. E seja:

$$P_{A_{si}} = \{a : a \in A_{S_i} \ / \ t_a = \text{parametrizado}\}$$

o subconjunto de A_{S_i} dos adaptadores parametrizados ativados pela situação de contexto s_i.

Para cada adaptador parametrizado $a_j \in P_{A_{si}}$, chama-se B_{a_j} o conjunto de atributos de contexto utilizado por esse adaptador:

$$B_{a_j} = \{b \ : \ b \ \text{ é atributo de contexto utilizado pelo adaptador } \ a_j \in P_{A_{si}}\}$$

Neste algoritmo, para identificar as adaptações comuns que podem ser realizadas para um grupo de clientes, não basta apenas avaliar se esses clientes possuem estados de contexto comuns. Para cada adaptador parametrizado, haverá nova divisão do grupo de clientes levando em consideração a igualdade dos valores correntes dos atributos de contexto (usados pelo adaptador).

Define-se B_{a_j,c_k} como o conjunto de valores correntes dos atributos de contexto B_{a_j} para o cliente c_k.

$$B_{a_j,c_k} = \{b_{c_k} : \ \text{atributo } b \in B_{a_j} \ / \ b_{c_k} = \ \text{valor corrente do atributo } b \text{ para } c_k\}$$

Para cada situação de contexto s_i, dois clientes c_k e c_m poderão estar no mesmo grupo, ou seja, realizar uma adaptação comum see:

$$e_{c_k,s_i} = e_{c_m,s_i} \ \wedge \ e_{c,s_i} = \text{ativo} \implies (\forall a_j \in P_{A_{si}}) \ (B_{a_j,c_k} = B_{a_j,c_m})$$

Ou seja, os clientes estão no mesmo grupo se seus estados de contexto na situação s_i forem iguais e, além disso, se o estado for ativo (i.e, haverá adaptação) então os atributos de contexto utilizados por adaptadores parametrizados devem necessariamente ter valores iguais.

O Algoritmo 3 descreve os passos para a execução de adaptações parametrizadas por contexto, para grupos de clientes.

O Algoritmo 3 utiliza o método auxiliar `ExecuteParametrizedAdapters`, que é apresentado no Algoritmo 4. Este método é responsável por aplicar uma seqüência de adaptações sobre uma mensagem, podendo os adaptadores serem parametrizados ou não. Desta forma ele também é responsável por repartir o grupo de clientes de acordo com o valor corrente de seus atributos de contexto. O método espera como entrada um grupo de clientes que estejam todos com a situação de contexto ativa, e portanto haverá adaptação de mensagem para todos os clientes.

Algoritmo 3: Adaptação de conteúdo utilizando adaptadores parametrizados, para um grupo de clientes

Entrada:
G_C - grupo de clientes para o qual a mensagem deve ser enviada
msg - mensagem a ser enviada, e adaptada se necessário
i - índice da condição de contexto analisada, $i = 1 \ldots |S|$
Resultado: mensagem adaptada e entregue aos clientes

AdapterMngr (G_C, msg, i)
if $i > |S|$ **then**
Send(msg, G_C) `/* Envia mensagem adaptada ao grupo de clientes` G_C `*/`
return
end
`/* Verificar quais clientes de` G_C `estão com a situação de contexto` i `ativa e dividí-los em dois grupos: (i) clientes com situação` i `ativa e (ii) clientes com condição não ativa */`
$G_{Si} \longleftarrow \{c_k : c_k \in G_C \ / \ e_{c_k,S_i} = \text{ativo}\}$
$G_{\neg Si} \longleftarrow \{c_k : c_k \in G_C \ / \ e_{c_k,S_i} = \text{inativo}\}$
if $G_{Si} \neq \oslash$ **then**
`/* Verificar se há adaptadores parametrizados para esta situação de contexto. */`
$P_{A_{si}} \longleftarrow \{a : a \in A_{S_i} \ \wedge \ t_a = \text{parametrizado}\}$
if $P_{A_{si}} = \oslash$ **then**
`/* Aplicar adaptações ativadas por` S_i `*/`
$msg' \longleftarrow$ ExecuteAdapters(A_{Si}, msg)
`/* Analisar próxima situação de contexto */`
AdapterMngr($G_{Si}, msg', i + 1$)
else
`/* Tratar adaptações parametrizadas separadamente */`
ExecuteParametrizedAdapters($G_{Si}, msg, i, A_{Si}, 1$)
end
end
if $G_{\neg Si} \neq \oslash$ **then**
`/* Executar o algoritmo para próxima situação de contexto */`
AdapterMngr($G_{\neg Si}, msg, i + 1$)
end
end

Lembrando que G_{Si} indica o grupo de clientes com o estado da condição de contexto s_i ativo, será chamado de G_{B_{aj},S_i} o grupo de clientes com a situação s_i ativa e cujos valores dos atributos de contexto B_{a_j} são iguais, quando a_j é um adaptador parametrizado. O conjunto de grupos G_{B_{aj},S_i} é chamado de G_{B,S_i}.

Note, também, que a ordem de aplicação dos adaptadores de cada A_{Si} não deve ser alterada.

O Algoritmo 4 é recursivo em relação à seqüência de adaptadores A_{Si}, dividindo o grupo de clientes quando o adaptador for parametrizado. Além disso, note que este algoritmo é uma recursão dentro do outro algoritmo recursivo (Algoritmo 3) que itera na seqüência de situações de contexto S.

Semelhante ao Algoritmo 2, a complexidade do Algoritmo 3 em relação ao número de adaptadores executados também varia entre $O(d)$ e $O(d \times k)$. Entretanto, a utilização de adaptadores parametrizados reduz o tamanho dos grupos com características comuns (existindo maior número de clientes isolados, ou grupos de tamanho 1) o que acaba por diminuir o ganho em relação ao Algoritmo 2, pois neste caso, os clientes só podem usufruir da mesma adaptação se estiverem com as mesmas situações de contexto ativas e possuírem os mesmo valores correntes dos atributos de contexto que são utilizados pelos adaptadores parametrizados.

4.4 Resumo e Conclusões

Neste capítulo, foi apresentada uma arquitetura de referência para serviços de disseminação de contexto em ambientes móveis que provê suporte a adaptações sensíveis ao contexto corrente dos clientes e comunicação publish/subscribe.

Além disso, foi mostrado o problema de escalabilidade devido ao elevado custo das adaptações de conteúdo, principalmente quando efetuadas para um elevado número de clientes. Para atenuar esse problema, foram propostos dois algoritmos, um para adaptações parametrizadas por contexto e outro para adaptações não parametrizadas, que exploram a criação de grupos de clientes com contexto similares.

Mostrou-se que a eficiência dos algoritmos depende principalmente do estado de contexto corrente dos clientes, apresentando maiores ganhos quanto maior for o grau de similaridade dos contextos entre os clientes. Pode-se observar, por exemplo, que a utilização de adaptadores parametrizados reduz o número de clientes com características comuns, o que reduz o ganho do algoritmo.

Outro ponto a se notar é que os algoritmos supõem que adaptaçãoes

estão em uma sequência não comutativa, ou seja, a ordem de execução dos adaptadores não pode ser alterada. Essa ordem pode ser definida pelas regras de adaptação. Outra alternativa para gerar essa sequência não comutativa seria incluir, em uma fase anterior a execução dos adaptadores, uma função que reordenasse a sequência de forma a otimizar sua execução (por exemplo, executando primeiro as adaptações de compressão). Além disso, considera-se que as adaptações não devam ser executadas em paralelo, pois como elas alteram o conteúdo, cria-se relação de dependência entre o resultado de uma adaptação prévia do conteúdo e conteúdo que será repassado para o adaptador posterior.

Algoritmo 4: Divisão de grupos de acordo com adaptadores parametrizados por contexto e execução das adaptações

Entrada:
G - grupo de clientes com situação i ativa
msg - mensagem a ser enviada, e adaptada se necessário
i - índice da condição de contexto analisada
A_{Si} - conjunto de adaptadores ativados pela situação corrente
j - adaptador avaliado
Resultado: mensagem adaptada para cada grupo

ExecuteParametrizedAdapters (G, msg, i, A_{Si}, j)
  if $j > |A_{Si}|$ **then**
    /* Terminaram as adaptações da situação de contexto i, continuar para as demais */
    AdapterMngr$(G, msg, i+1)$
    return
  end
  /* Se o adaptador corrente (a_j) for parametrizado, dividir G em grupos de clientes para os quais os valores dos atributos de contexto usados pelo adaptador a_j estejam iguais */
  $a_j \longleftarrow A_{Si}[j]$
  if t_{a_j} = *parametrizado* **then**
    $B_{a_j} \longleftarrow$ {b : b é atributo de contexto utilizado pelo adaptador a_j}
    /* Obter os valores dos atributos de contexto (utilizados por a_j) de cada cliente */
    $B_{a_j,c_k} \longleftarrow$ {x: x é valor corrente de cada atributo de contexto de B_{a_j} para o cliente c_k} , $\forall c_k \in G$
    $B_{a_j,C} \longleftarrow \bigcup B_{a_j,c_k}$
    /* Verificar os clientes com valores de contexto iguais, e separá-los em grupos. G_{B,S_i} é a lista desses grupos */
    $G_{B,S_i} \longleftarrow$ SeparaGrupos $(G, B_{a_j,C})$
    foreach *grupo* $G_{(B_{aj},S_i)} \in G_{B,S_i}$ **do**
      $msg' \longleftarrow$ ExecuteAdapter (a_j, msg)
      /* Executar a próxima adaptação para este grupo */
      ExecuteParametrizedAdapters$(G_{(B_{aj},S_i)}, msg', i, A_{Si}, j+1)$
    end
  end
  else /* a_j não parametrizado */
    $msg' \longleftarrow$ ExecuteAdapter (a_j, msg)
    /* Executar próxima adaptação da seqüência */
    ExecuteParametrizedAdapters$(G, msg', i, A_{Si}, j+1)$
end