Adaptação de Conteúdo Sensível a Contexto para Dispositivos Móveis em Sistemas Publish/Subscribe

Hana Karina S Rubinsztejn, Markus Endler [1]

[1]Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rua Marquês de São Vicente, 225
22453-900, Rio de Janeiro,RJ, Brasil

{hana,endler}@inf.puc-rio.br

Abstract. *Services for information dissemination ("push" services) are being widely used, in particular for applications involving mobile users, making it necessary to adapt the spread out content individually and dinamically for each customer. Asynchronous communication of the type publish/ subscribe (P/S) is considered the most appropriate for this type of service. On the other hand, systems for context-aware content adaptation do not support this type of communication. In this article we present an architecture for publish/subscribe systems with context-aware content adaptation, that uses an algorithm that optimizes the content adaptation for large sets of customers.*

Resumo. *Serviços para disseminação de informações (serviços "push") têm sido amplamente utilizados, em particular para aplicações envolvendo usuários móveis, o que torna necessário adaptar o conteúdo difundido dinamica- e individualmente para cada cliente. Comunicação do tipo publish/subscribe (P/S) é considerada por muitos como a mais apropriada para este tipo de serviço. Por outro lado, sistemas para adaptação de conteúdo sensível a contexto não dão suporte a este tipo de comunicação, até porque a adaptação precisa ser individual, e portanto conflita com o paradigma de comunicação um-para-muitos. Neste artigo apresentamos uma arquitetura para sistemas publish/subscribe com adaptação sensível a contexto, que utiliza um algoritmo que otimiza a adaptação de conteúdo para grandes conjuntos de clientes de uma difusão.*

1. Introdução

Nos últimos anos, a evolução das tecnologias sem fio tem impulsionado e motivado o desenvolvimento de aplicações capazes de explorar de forma vantajosa as características da mobilidade do usuário. Além das características primárias que fazem parte da lógica de negócio, essas aplicações levam em conta o contexto corrente do usuário, do dispositivo ou da rede sem fio para prover serviços mais adequados ao usuário final. Como exemplo de uso de contexto, essas aplicações poderiam considerar a localização do usuário ou a vazão do enlace da rede sem fio para personalizar o seu comportamento, como, por exemplo, enviar uma oferta de um produto de uma loja próxima ao usuário ou implementar uma compressão dos dados transmitidos para amenizar algum problema da rede.

Serviços e middleware de provisão de contexto [Chen 2005, Dey et al. 2001] têm sido extensivamente investigados e propostos com o objetivo de auxiliar o desenvolvimento de aplicações sensíveis ao contexto, especialmente para redes móveis. Informações

de contexto podem ser relativas a fatores físicos, como ambiente (luz, temperatura, ...), à infra-estrutura computacional (recursos do dispositivo e da rede) e localização, e a fatores humanos como preferências, atividade, relacionamento social,etc.

De fato, o acesso às informações contextuais dos dispositivos e usuários abre uma grande variedade de novas possibilidades para implementar aplicações distribuídas para usuários móveis. Entretanto nas redes móveis um fator de complexidade adicional diz respeito à heterogeneidade dos dispositivos usados, e das tecnologias e protocolos de comunicação utilizados. As aplicações precisam tratar dispositivos clientes com características distintas, e em contextos variados. Desta forma, é interessante que o conteúdo transmitido pela aplicação, por exemplo, uma página HTML, uma mensagem ou uma imagem, seja personalizado para cada cliente, para atender às características específicas de cada um, como por exemplo, uma formatação/codificação adequada ao tamanho da tela e/ou ao número de cores disponíveis em seu dispositivo.

É comum utilizar a adaptação estática de conteúdo [Noble et al. 1995, Mohan et al. 1999], na qual são mantidas várias cópias do conteúdo em diferentes versões para atender a cada tipo de cliente. Nesta abordagem estática, o servidor apenas seleciona a versão mais adequada (para apresentação) de acordo com a situação (contexto, em geral relacionado ao tipo de tela) do dispositivo. Esse tipo de solução não é escalável e torna-se inviável à medida que aumenta o volume total de conteúdo, pois gera um alto custo de desenvolvimento (autoria de várias versões) e de manutenção, sendo necessário criar uma nova versão de cada conteúdo para cada nova situação (ou dispositivo) que se deseja atender. Além disso, o número de contextos (situações) que podem ser satisfatoriamente atendidos fica limitado ao número de versões do conteúdo existentes.

Outra abordagem é adaptação dinâmica de conteúdo [Fox et al. 1998, Chandra et al. 2000]. Nesta abordagem há apenas uma versão do conteúdo (original) e no momento do envio deste para o cliente a adaptação adequada é efetuada de acordo com o contexto corrente do cliente. Esta solução apresenta um custo de desenvolvimento e manutenção menor que a anterior, além de permitir um conjunto mais amplo e dinâmico de contextos (relacionados à rede, ao dispositivo, ao usuário, etc). Para a adaptação de conteúdo, várias técnicas podem ser utilizadas, dentre elas destilação, refinamento, sumarização, filtragem e transcoding. Uma variante desta abordagem é a utilização de cache para armazenar adaptações já realizadas. Entretanto, esta solução torna-se custosa e pouco útil quando há grande variedade de contextos e quando o conteúdo disseminado é dinâmico.

Adaptações de conteúdo geralmente são operações custosas e demandam um alto poder de processamento, principalmente para conteúdo multimídia. Efetuar todas estas operações no servidor, para cada cliente móvel, torna-se pouco eficiente e escalável. Assim, uma abordagem comum na arquitetura de aplicações para redes sem fio é a utilização de intermediários [Lum and Lau 2002, Chen et al. 2003], que intermedeiam toda a comunicação entre clientes e servidores, e são responsáveis por realizar transformações, adaptações ou funções de gerência em benefício de um ou mais clientes, tais como adaptação de conteúdo, tradução de protocolos, buffering de mensagens, gerência de handover, etc. Em geral, proxies estão localizados em nós da rede fixa, e permitem o desenvolvimento de "clientes magros".

Devido às características do enlace sem fio, a comunicação assíncrona do tipo publish/subscribe (P/S) é considerada por muitos como a mais apropriada para computação móvel [Cugola and Jacobsen 2002], onde os dispositivos podem estar freqüentemente indisponíveis ou desconectados. Publishers e subscribers são desacoplados, interagindo assincronamente e sem a necessidade de estarem simultaneamente ativos para a troca de mensagens. Fica a cargo da infraestrutura P/S o armazenamento e distribuição das mensagens.

Entretanto os sistemas adaptativos baseados em proxies não provêm suporte para esse tipo de comunicação no nível da aplicação, permitindo apenas a comunicação síncrona (request/reply).

Um desafio em relação à comunicação P/S em sistemas para adaptação sensível a contexto é o fato de que a comunicação 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. Entretanto, a adaptação depende do contexto corrente de cada cliente e, portanto, deve ser individualizada. Para permitir uma comunicação assíncrona P/S com adaptação sensível a contexto, o proxy seria responsável por efetuar a interceptação das mensagens e replicação destas para aplicar as adaptações específicas para cada cliente que as necessite. Neste caso, para uma maior eficiência, é preciso também desenvolver um mecanismo para otimizar as adaptações (ou seja, para reduzir a replicação de mensagens), como por exemplo, a criação de grupos de clientes com contextos semelhantes, onde a replicação e adaptação da mensagem não seriam para cada cliente, mas sim para cada grupo. Como os sistemas adaptativos atuais não provêm suporte a comunicação P/S, nosso objetivo é desenvolver um sistema capaz de atender a esta necessidade, levando em consideração questões de eficiência e escalabilidade.

O artigo está estruturado como segue. A seção 2 descreve uma arquitetura para suporte à adaptação personalizada em sistemas de comunicação P/S. Na seção 3 apresentamos formalmente o problema e uma proposta de solução, com um método para aumento da eficiência. Na seção 4 são apresentados alguns trabalhos existentes e as conclusões são apresentadas na seção 5.

2. Arquitetura de sistemas P/S com adaptação de conteúdo personalizada

Apresentamos na Figura 1 a arquitetura proposta, para aplicações móveis sobre sistemas publish/subscribe, que provê adaptação individualiza segundo o contexto de cada cliente.

A arqitetura é formada logicamente por três camadas: (*i*) uma camada base, composta do middleware comunicação publish/subscribe e do middleware de provisão de contexto, (*ii*) uma camada intermediária responsável pela gerência de adaptações baseada no contexto corrente dos clientes , e (*iii*) uma camada de aplicação. Estes componentes da arquitetura são descritos com mais detalhes a seguir.

Middleware de comunicação P/S : é utilizado como base para disseminação de informações (conteúdo) entre os clientes da aplicação. Deve permitir subscrições baseadas em tópicos ou conteúdo, e é responsável por prover suporte a mobilidade dos usuários. Além disso, é interessante que tenha uma arquitetura distribuída para melhor escalabilidade.

Para a interação com a gerência de adaptações, o middleware P/S deve prover

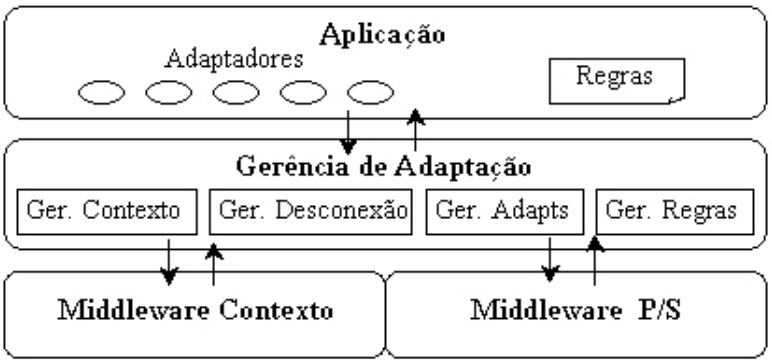

Figura 1. Arquitetura para sistemas P/S com adaptação sensível a contexto

meios de interceptação da notificação de publicações, permitindo: *i*) a recuperação do conjunto de clientes destinatários de uma notificação, e *ii*) a publicação para subconjuntos dos clientes destinatários originais, sendo o conteúdo da notificação alterado.

Middleware de provisão de contexto: responsável por coletar, inferir e distribuir as informações de contexto de interesse da aplicação. Ele se comunica com a camada de gerência de adaptação através de eventos que informam a ocorrência de um contexto (para ativação da adaptação correspondente) e eventos no momento em que o contexto deixa de ocorrer (para desativação da adaptação). Além disso, o middleware deve permitir a requisição de eventos de contextos compostos definidos por expressões que combinem diversas informações básicas (como conectividade, características de dispositivo e nível de energia). Ele também é responsável por informar desconexão (permanente ou temporária) de dispositivos.

Gerência de adaptações: responsável por relacionar os contextos de interesse da aplicação com as adaptações desejadas (ou seja, as regras de adaptação) ativando/desativando as mesmas quando necessário.É composta por uma série de componentes: gerente de contexto, gerente de adaptadores, gerente de regras de adaptação, e gerente de desconexão.

O gerente de contexto é responsável por atualizar o estado (contexto) dos clientes. Para isso, obtém os diferentes contextos de interesse da aplicação para cada cliente, registrando-se no middleware de contexto, e atualizando o estados destes clientes quando da recepção de uma notificação de mudança de contexto.

O gerente de regras de adaptação requisita as regras para adaptação (contextos que disparam as adaptações) da camada de aplicação. O núcleo deste módulo é a máquina de decisão, que é responsável por verificar o contexto corrente do cliente e indicar a cadeia de adaptadores a ser utilizada. A máquina de decisão é acionada sempre que há o envio de uma mensagem para o cliente, e é neste momento que se avalia o contexto do cliente e ativa-se ou não os adaptadores de conteúdo da mensagem.

O gerente de adaptadores é responsável por carregar os adaptadores desenvolvidos pela aplicação e é encarregado também pela execução da cadeia de adaptadores selecionada pela máquina de decisão.

O gerente de desconexão requisita informações de desconexão do gerente de con-

texto, e pode na ocorrência de desconexões (voluntárias ou não), ativar um caching de mensagens para o cliente desconectado. A política de caching adotada deve ser definida e implementada pelo desenvolvedor da aplicação.

Aplicação: responsável por definir e implementar as adaptações específicas para atender suas necessidades. Além disso, deve configurar as regras que relacionam os contextos de interesse que ativam as adaptações. Também pode definir regras para momentos de desconexão que ativam o armazenamento das informações, e quais políticas de caching devem ser utilizadas nestes momentos.

3. Adaptação de conteúdo em sistemas Pub/Sub

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.

Em aplicações adaptativas sensíveis a contexto, a personalização de conteúdo das mensagens deve ser realizada de acordo com o contexto de cada cliente. O desafio é permitir este tipo de personalização em uma comunicação publish/subscribe.

Para a entrega personalizada 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. Esta solução é dispendiosa e pouco eficiente, e no caso de adaptações que demandam muitos recursos computacionais pode tornar o sistema não escalável.

Nosso objetivo é efetuar as adaptações de forma a otimizar (reduzir) o custo. 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 sub-grupos de clientes com características comuns de acordo com suas condições atuais de contexto, nos quais o mesmo conjunto de adaptadores é utilizado.

A seguir formalizamos o problema e a nossa solução, apresentando um algoritmo para gerenciar as adaptações de forma eficiente. Este algoritmo é utilizado na implementação da máquina de decisão do gerente de regras, da camada de Gerência de Adaptações discutido na seção anterior.

3.1. Algoritmo de gerência execução de adaptações

Adotamos a abordagem de adaptação dinâmica de conteúdo na qual , como já mencionado, a adaptação é executada apenas no momento do envio do conteúdo para o cliente, de acordo com o contexto corrente do cliente.

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

Denotamos a seqüência das n condições de contexto de interesse de:

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

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, há um subconjunto de adaptadores de A que são ativados por esta condição em uma determinada ordem. Denotamos $s_i \longrightarrow a$, a situação de contexto s_i que dispara a adaptação a. A seqüência de adaptadores ativados por s_i, A_{Si}, é definida por:

$$A_{Si} = (a : a \in A \wedge s_i \longrightarrow a) \;\; s_i \in S,\, i = 1 \ldots n, \tag{2}$$

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.

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

O estado das condições de contexto de todos os clientes de C no momento da chegada da mensagem ao proxy 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,

$$\begin{gathered} E_{c_k} =< e_{c_k,s_1}, e_{c_k,s_2}, \ldots, e_{c_k,s_n} >, \;\; e_{c_k,s_i} \in \{\text{ativo}\,;\, \text{inativo}\} \\ E_C = \cup E_{c_k},\; c_k \in C \end{gathered} \tag{3}$$

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 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.

Denotamos 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 seqüências para todos os clientes chamamos de A_C. Assim:

$$\begin{gathered} A_{c_k} = A_{Si} \oplus A_{Sj} \oplus \ldots \;\; \forall s_i \in S \;\text{ tq }\; e_{c_k,s_i} = \text{ ativo} \\ A_C = \cup A_{c_k},\; \forall\, c_k \in C \end{gathered} \tag{4}$$

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. Chamamos "prefixos" Pr estas adaptações comuns, 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, o proxy efetua o processo descrito a seguir:

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

$$A_{c_k} = (\oplus A_{Si},\ \forall s_i \in S\ \text{tq}\ e_{c_k,s_i} = \text{ativo})$$

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 possuirem prefixo comum. Cada prefixo é definido por:

$$Pr = \ (a : a \in A_{c_i} \cap A_{c_j}\ \wedge\ a \text{ aparece na mesma ordem em ambas seqüências})\ (\forall c_i, c_j \in C);$$

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

Descrevemos os passos acima no Algoritmo 1, iterando 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. Definimos 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. Assim:

$$G_{Si} \longleftarrow \{c_k : c_k \in G_C\ \wedge\ e_{c_k,s_i} = \text{ativo}\},\ \forall s_i \in S$$
$$G_{\neg Si} \longleftarrow \{c_k : c_k \in G_C\ \wedge\ e_{c_k,s_i} = \text{inativo}\},\ \forall s_i \in S$$

Inicialmente, os argumentos de invocação do Algoritmo 1 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:

$$AdtapterMngr(C, m, 1)$$

Uma observação é que no Algoritmo 1 o método `Send` é responsável por enviar a mensagem ao grupo de clientes especificado, representando aqui a interface com o middleware de P/S.

3.1.1. 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, vamos considerar que cada situação de contexto torne ativo apenas um adaptador. Os parâmetros considerados são:

- Clientes: $C = \{c_1, c_2, ..., c_6\}$
- Condições de contexto: $S = (s_1, s_2, ..., s_5)$.
- Adaptadores: $A = \{A_1, A_2, ..., A_5\}$
- Adaptadores ativados por cada situação de contexto:

Algoritmo 1: Adaptação de conteúdo 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 */
    return
  /* 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 ⟵ {c_k : c_k ∈ G_C ∧ e_{c_k,s_i} = ativo}
  G_¬Si ⟵ {c_k : c_k ∈ G_C ∧ e_{c_k,s_i} = inativo}
  if G_Si ≠ ∅ then
    /* Aplicar adaptações referentes a situação de
       contexto S_i                                     */
    msg' ⟵ ExecuteAdapters(A_Si, msg)
    /* Executar o algoritmo, sobre a mensagem
       adaptada, para próxima situação de contexto      */
    AdapterMngr(G_Si, msg', i + 1)
  if G_¬Si ≠ ∅ then
    /* Executar o algoritmo para próxima situação de
       contexto                                         */
    AdapterMngr(G_¬Si, msg, i + 1)
end
```

– $A_{S_1} = (A_1)$, $A_{S_2} = (A_2)$, $A_{S_3} = (A_3)$, $A_{S_4} = (A_4)$, $A_{S_5} = (A_5)$.

Por exemplo, para uma aplicação que transmitisse as transparências (composta de textos e imagens) de uma palestra, as condições dos clientes móveis poderiam ser: conexão ruim (s_1), em movimento rápido (s_2), energia inferior a 10% (s_3), luminosidade elevada (s_4) e compatibilidade do formato transmitido (s_5). 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 (ex jpeg para tiff), respectivamente.

A Tabela 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 caracter '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.

Tabela 1. Seqüências A_{C_k} para o exemplo

	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	

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

Para melhor demonstrar a formação de grupos, a replicação da mensagem apenas nos momentos realmente necessários e a aplicação dos adaptadores, a Figura 2 apresenta a árvore de divisão de grupos e fluxo da adaptação da mensagem para o exemplo anterior.

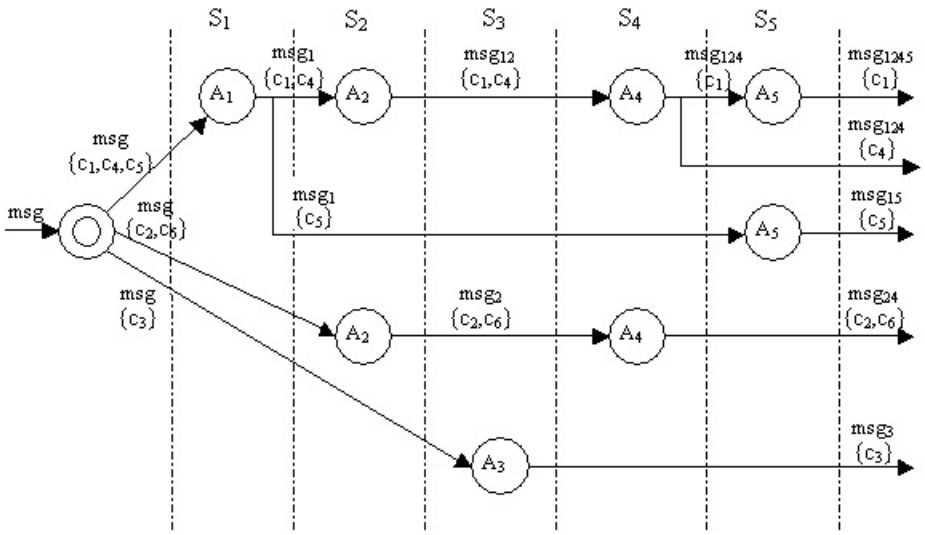

Figura 2. Envio de mensagem a um grupo pub/sub

Com a utilização do algoritmo proposto (Algoritmo 1), podemos perceber que, neste exemplo, o número de adaptações é reduzido de 14 para 8 (redução de aproximadamente 40%) e são criadas 5 mensagens distintas.

3.1.2. Avaliação do algoritmo

Como dito anteriormente, o Algoritmo 1 tem como objetivo reduzir o número total de adaptações realizadas, que seria, no caso do algoritmo mais simples, da ordem de número de clientes multiplicado pelo número de adaptadores. Esta otmização (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 estes prefixos.

Denominamos 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, definimos:

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

Para calcularmos a complexidade do Algoritmo 1 em relação ao número de adaptações realizadas, definimos b 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 estados e número de clientes em cada grupo:

$$T(n,k) \quad = \quad f(S_1) + T(n-1, \tfrac{k}{b}) + T(n-1, k - \tfrac{k}{b})$$

Mostraremos que o Algoritmo 1 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:

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

Note que no melhor caso a complexidade do Algoritmo 1 se iguala à complexidade de realizar adaptações para apenas um cliente.

O Algoritmo 1 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 é quando todos os ramos da recursão são executados. Isso ocorre sempre que o grupo de clientes é repartido ao meio, ou seja, quando $b = 2$. Neste caso, a equação de recorrência pode ser escrita como:

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

Sabendo que $T(0,k) = 0,\ \forall\, k$ e como $f(S_i) = 0$ ou $f(S_i) = |A_{Si}|$, então $f(S_i) \leq |A_{Si}|$ e assim:

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

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

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

Sabemos que cada grupo de clientes na execução possui pelo menos um cliente, ou seja, $k/2^n \geq 1$, portanto $2^n \leq k$. Assim temos, o pior caso:

$$T(n,k) < d \times k$$

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 cliente multiplicado pelo número de clientes: $T(n,k) = O(d \times k)$.

Entretanto na maioria dos casos, 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 sub-grupo 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 poucas dezenas. Portanto, nestes casos sempre haverá grupos de clientes com as mesmas situações, ou seja, clientes com necessidades de adaptações em comum, que serão executadas apenas uma única vez para todo o grupo e portanto, $T(n,k) << d \times k$.

4. Trabalhos relacionados

Apresentamos nesta seção alguns trabalhos relacionados ao nosso sistema publish/subscribe com suporte à adaptação de conteúdo sensível a contexto de clientes móveis.

Sistemas publish/subscribe para redes móveis, como Jedi [Cugola and Jacobsen 2002] e Siena [Caporuscio et al. 2003], em geral apenas tratam de problemas de mobilidade dos clientes e de roteamento eficiente para requisições baseadas em conteúdo. Entretanto não se preocupam com o tratamento/adaptação do conteúdo devido a mudanças de contexto. Além disso, tais sistemas poderiam ser utilizados como middleware P/S em nossa arquitetura, desde que fornecem as interfaces de interceptação de publicações.

Com relação aos sistemas de adaptação sensíveis a contexto, estes apenas permitem a comunicação request/reply. Eles não suporte à comunicação P/S, e nem verificam as conseqüências/dificuldades que surgem deste tipo de comunicação.

Várias arquiteturas para adaptação têm sido desenvolvidas tais como [McKinley et al. 2003, Brewer and et al. 1998, Angin et al. 1998, Ardon et al. 2003, Chuang et al. 2004], seja permitindo personalização ou extensão para resolução de um problema em particular.

Assim como a maioria destes sistemas, o nosso permite a criação de cadeias de adaptadores, de acordo com o contexto dos clientes. Entretanto, nós também permitimos, na especificação das políticas de adaptação, a composição de contextos complexos, e não apenas de contextos simples (como perfis dispositivo e cliente, e aspectos do enlace sem fio) como os demais. Além disso, apenas a nossa arquitetura provê suporte a desconexão (permanente), oferecendo mecanismos de buffering de mensagens sensíveis à conectividade, com políticas programadas pelo desenvolvedor da aplicação. E considerando aspectos de comunicação, nossa arquitetura é a única a oferecer suporte à comunicação assíncrona do tipo publish/subscribe, permitindo adaptações personalizadas.

5. Conclusão

Vimos que em aplicações ”push”para redes móveis que adaptam conteúdo de acordo com o contexto corrente dos clientes, o desempenho da difusão pode ser afetado à medida que o número de clientes aumenta, dado que muitas adaptações demandam altos custo de processamento e armazenamento. Conseqüentemente a escalabilidade do sistema fica prejudicada. Para tratar este problema apresentamos um método que agrupa os clientes com contextos similares, de tal forma a aplicar as adaptações para estes grupos ao invés de aplicá-las individualmente. Este método reduz o número de adaptações, e portanto torna o sistema mais eficiente.

Um protótipo da arquitetura proposta (vide na secao 2) já foi desenvolvido, utilizando como middleware para provisão de contexto o serviço CIS (Context Information Service) da MoCA (Mobile Collaboration Architecture) [Sacramento et al. 2004]. Além disso, para a comunicação publish/subscribe foi utilizado outro componente da MoCA o ECI (Event-Based Communication Interface).

A camada de Gerência de Adaptações foi desenvolvida como um framework [1]

[1]Maiores detalhes sobre o framework podem ser obtidos em [Rubinsztejn et al. 2005]

cujos pontos de extensão e configuração são os adaptadores de conteúdo e políticas de caching, e a definição das regras de adaptação via arquivos XML. Na versao atual do protótipo, o Gerente de Adaptações implementa apenas o algoritmo ingênuo, no qual as mensagens são replicadas para cada cliente da notificação P/S, e as adaptações são aplicadas individualmente.

Algumas aplicações já foram implementadas utilizando este framework. Pudemos observar que adaptadores de imagens são realmente bastante custosos. Por exemplo um adaptador para reduzir a resolução de uma imagem de 1280x1024 pixels para uma imagem 240x320 (por exemplo, para adequar uma imagem ao tamanho de tela de um cliente HP iPAQ Hx2490) demora em média 2s. Assim, percebemos que, a repetição desta adaptação para um número grande de clientes (em uma notificação P/S) usando o algoritmo ingênuo, é bastante dispendiosa e torna o processamento do intermediário bem mais lento. Isto mostrou-nos a necessidade de implementar uma otimização usando a idéia de grupos de clientes.

Uma versão otimizada do gerente de adaptações, utilizando o algoritmo com gerência de grupos proposto na seção 3 já foi implementado e está sendo testado. Acreditamos que em breve teremos dados concretos que comprovam o melhor desempenho da versão otimizada. No entanto, deve-se dizer que o real benefício da versão otimizada só ficará evidente em testes (simulados) com muitos clientes cujos contextos estejam no mesmo estado. Além disso, os testes devem indicar precisamente a sobrecarga gerada pela gerência dinâmica dos grupos.

O algoritmo descrito na seção 3 não considera a existência de adaptadores parametrizados por contexto. Adaptadores parametrizados por contexto são adaptadores que não apenas são acionados por condições de contexto, mas também utilizam os valores correntes de atributos de contexto como parâmetros para a execução da adaptação. Já desenvolvemos um algoritmo para o gerente de adaptações lidar com este este tipo de adaptadores. Este algoritmo em breve será incorporado ao protótipo para ser testado. Infelizmente, este algoritmo tende a ser um pouco menos eficiente do que o atual, mas indispensável para o correto tratamento de adaptadores parametrizados por contexto.

Referências

Angin, O., Campbell, A., Kounavis, M., and Liao, R.-F. (1998). The Mobiware Toolkit: Programmable Support for Adaptive Mobile Netwoking. *IEEE Personal Communications Magazine, Special Issue on Adapting to Network and Client Variability*.

Ardon, S., Gunningberg, P., LandFeldt, B., Y. Ismailov, M. P., and Seneviratne, A. (2003). March: a distributed content adaptation architecture. *International Journal of Communication Systems, Special Issue: Wireless Access to the Global Internet: Mobile Radio Networks and Satellite Systems.*, 16(1).

Brewer, E. and et al. (1998). A network architecture for heterogeneous mobile computing. *IEEE Personal Communications Magazine*.

Caporuscio, M., Carzaniga, A., and Wolf, A. L. (2003). Design and evaluation of a support service for mobile, wireless publish/subscribe applications. Technical Report CU-CS-944-03, Department of Computer Science, University of Colorado.

Chandra, S., Ellis, C., and Vahdat, A. (2000). Differentiated multimedia web services using quality aware transcoding. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 961–969, Tel Aviv, Israel.

Chen, H. (2005). *An Intelligent Broker Architecture for Context-Aware Systems*. PhD thesis, University of Maryland, Department of Computer Science and Electrical Engineering.

Chen, Y.-F., Huang, H., Jana, R., Jim, T., Hiltunen, M., John, S., Jora, S., Muthumanickam, R., and Wei, B. (2003). imobile ee: An enterprise mobile service platform. *Wireless Networks*, 9(4):283–297.

Chuang, S. N., Chan, A. T., Cao, J., and Cheung, R. (2004). Actively deployable mobile services for adaptive web access. *IEEE Internet Computing*, 08(2):26–33.

Cugola, G. and Jacobsen, H.-A. (2002). Using publish/subscribe middleware for mobile systems. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):25–33.

Dey, A., Salber, D., and Abowd, G. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16.

Fox, A., Goldberg, I., Gribble, S. D., and Lee, D. C. (1998). Experience with top gun wingman: A proxy-based graphical web browser for the 3com palmpilot. In *IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98)*, pages 15–18, Lake District, UK.

Lum, W. and Lau, F. (2002). On balancing between transcoding overhead and spatial consumption in content adaptation. In *Proc. of ACM Mobicom 2002*, pages 239–250, Atlanta, USA.

McKinley, P. K., Padmanabhan, U. I., Ancha, N., and Sadjadi, S. M. (2003). Composable proxy services to support collaboration on the mobile internet. *IEEE TRANSACTIONS ON COMPUTERS*, 52(6):713–726.

Mohan, R., Smith, J., and Li, C.-S. (1999). Adapting multimedia internet content for universal access. *Multimedia, IEEE Transactions on*, 1(1):104–114.

Noble, B., Satyanarayanan, M., and Price, M. (1995). A programming interface for application-aware adaptation in mobile computing. In *MLICS '95: Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*, pages 57–66, Berkeley, CA, USA. USENIX Association.

Rubinsztejn, H. K. S., Endler, M., and Rodriguez, N. (2005). A framework for building customized adaptation proxies. *IFIP International Federation for Information Processing : Intelligence in Communication Systems*, pages 1–11.

Sacramento, V., Endler, M., Rubinsztejn, H., Lima, L., Goncalves, K., and Bueno, G. (2004). An architecture supporting the development of collaborative applications for mobile users. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004. WETICE 2004. 13th IEEE International Workshops on*, pages 109–114, University of Modena and Reggio Emilia, Modena, Italy.