6
Avaliação do Sistema

Este capítulo descreve os resultados da avaliação de desempenho e da avaliação qualitativa de uso do ProxyFramework. A primeira parte descreve alguns resultados da análise de desempenho do gerenciamento de adaptações, incluindo a obtenção do contexto dos clientes, checagem das regras de adaptação e execução das adaptações. A segunda parte deste capítulo apresenta alguns resultados de uma pesquisa sobre a utilização do ProxyFramework para desenvolvimento de algumas aplicações móveis adaptativas, feita com usuários (desenvolvedores) destas.

6.1
Avaliação de Desempenho

Esta seção descreve alguns testes de desempenho, realizados com o propósito de avaliar a escalabilidade e o tempo de resposta do sistema.

Para os testes utilizou-se os serviços da MoCA (descritos na Seção 2.4), em especial, o serviço de contexto (CIS) e uma aplicação de teste dividida em três partes: servidor, proxy e clientes.

A aplicação teste adota o paradigma de comunicação publish/subscribe: o servidor publica mensagens, de tempos em tempos, para um único tópico, chamado "Testes", e todos os clientes são assinantes (*subscribe*) deste tópico, recebendo todas as mensagens (imagens) publicadas pelo servidor. A imagem a ser transmitida e o intervalo de envio são parâmetros da aplicação servidora. O proxy é responsável pela comunicação com o CIS, através do qual obtém informações de contexto dos clientes, e também pelas adaptações de conteúdo nas imagens, de acordo com o estado de contexto de cada cliente.

A implementação do proxy é uma instância do ProxyFramework, descrito no Capítulo 5. As regras de adaptação do proxy são configuradas de acordo com o teste realizado, e os adaptadores de imagem são carregados conforme definido pelo arquivo XML de configuração do ProxyFramework. Além disso, em sua implementação, tem-se um pool de quatro threads para execução de todos os adaptadores, uma thread para o Gerente de Comunicação, responsável pelo envio ordenado das mensagens, e duas threads para os demais gerenciadores.

Cada gerenciador tem uma fila de espera, onde as mensagens ficam armazenadas enquanto aguardam atendimento por uma das threads disponíveis.

Conforme já mencionado na Seção 2.4, o CIS é responsável pelo envio dos eventos que caracterizam as situações de contexto solicitadas pelo proxy (configuradas no arquivo XML). Para isso, é preciso haver monitores enviando os dados de contexto de cada dispositivo cliente. Cada monitor é responsável por coletar as informações sobre atributos de contexto do dispositivo (uso de CPU, bateria, intensidade de sinal, etc) no qual está instalado, e por transmitir essas informações ao CIS periodicamente, por exemplo, a cada segundo. Ao receber tais informações o CIS infere a ocorrência de situações de contexto solicitadas pelo proxy e envia um evento em caso afirmativo.

Como o número de equipamentos móveis disponível é limitado, seria impossível realizar testes de desempenho de escalabilidade. Além disso, usando dispositivos móveis reais, as situações de contexto seriam aleatórias, dificilmente reproduzíveis, o que por sua vez, impediria a comparação entre os casos de teste dos diferentes algoritmos para gerenciamento de adaptações (discutidos na Seção 4.3). Por isso, foi utilizado o simulador de monitor (Monitor Simulator da MoCA) que é capaz de enviar dados relativos a um ou mais dispositivos clientes, conforme descrito na Seção 2.4. Dessa forma, foi possível realizar testes mais controlados, permitindo variar o número de clientes e controlar as condições de contexto dos mesmos.

Vale ressaltar que, apesar desta configuração não ser realista, os resultados de desempenho obtidos nesses testes não difeririam de uma situação em que os monitores executassem nos dispositivos clientes, já que estes monitores apenas interagem com o CIS, enviando os dados de contexto. Para o serviço CIS, a origem dos dados de contexto é transparente, sendo indiferente se estes são originários de um simulador ou um monitor real.

Nos testes, foram utilizadas estações de trabalho com a seguinte configuração: Pentium IV 2.8GHz, 512MB de RAM, executando WindowsXP Professional em uma rede local Fast-Ethernet. Uma estação executou exclusivamente o proxy, outra executou o serviço CIS e os simuladores de monitores, e uma terceira executou o servidor e os clientes da aplicação teste. Este esquema é ilustrado na Figura 6.1.

Para os testes de desempenho, utilizou-se AspectJ [123, 124] para a instrumentação do código do ProxyFramework com aspectos, que permitem a interceptação de chamadas a métodos, coletando seus parâmetros, alterando ou impedindo sua execução, além da inclusão de instruções que registram em arquivos de log o tempo de processamento de alguns desses métodos.

Inicialmente, foram realizados experimentos que medem o tempo médio

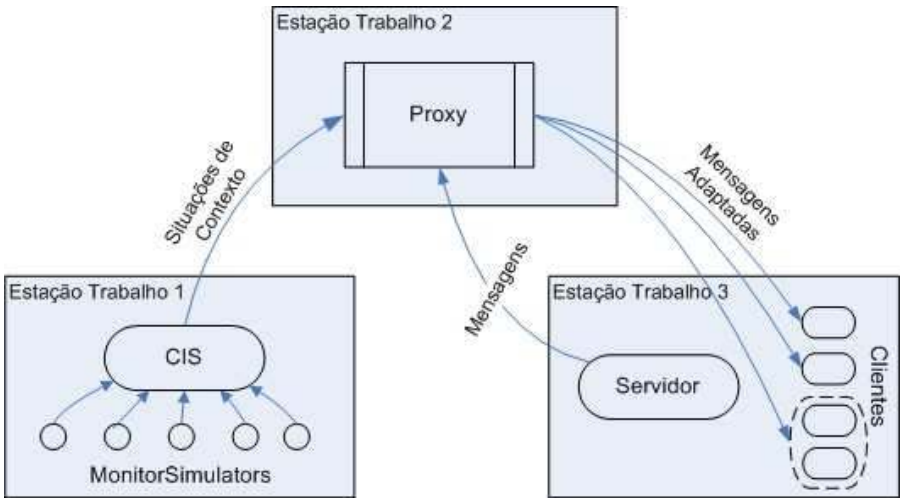

Figura 6.1: Esquema de execução dos testes.

de uma adaptação para diferentes tamanhos de mensagens para mostrar o quão significativo pode ser o custo (em tempo) de cada adaptação. Em seguida, a Seção 6.1.2 apresenta uma comparação entre os tempos de execução do algoritmo força bruta (Algoritmo 1) e do algoritmo *Context-Aware Client Grouping* (Algoritmo 2), discutidos na Seção 4.3. Este último algoritmo será chamado de "algoritmo com grupos"daqui em diante. Na Seção 6.1.3, é analisada a influência do número de regras de adaptação sobre o tempo de execução do algoritmo com grupos. Na Seção 6.1.4, avalia-se a execução deste algoritmo variando-se a quantidade de adaptações realizadas para cada cliente e, nas seções seguintes, esse experimento é realizado para o algoritmo força bruta e ingênuo, para fins de comparação. Finalmente, na Seção 6.1.7, é apresentado o comportamento do algoritmo com grupos para seu pior caso.

6.1.1 Custo de uma adaptação

Primeiramente, foi medido o tempo médio de uma adaptação, para os diferentes adaptadores de imagem disponibilizados atualmente para Proxy-Framework, variando-se o tamanho da imagem original.

Os adaptadores testados foram: *i*) Color-to-Gray, que converte uma imagem colorida para escala de cinza, *ii*) Reduce Quality, que converte a imagem para JPEG, alterando sua qualidade para o fator especificado, *iii*) Scale, que altera o tamanho da imagem de acordo com o fator desejado, e *iv*) Crop Center, que recorta a imagem ao centro no tamanho definido.

Os resultados apresentados na Tabela 6.1 correspondem à média de 20 execuções, com intervalo de confiança a 95%.

A tabela apresenta apenas o tempo total consumido por cada adaptação.

Tamanho Imagem (kB)	Tempo total (ms)			
	Color-to-Gray	Reduce Quality (40%)	Scale (60%)	Crop Center (100 x 150)
50	78.47	62.63	63.63	100.58
100	133.95	175.53	97.32	166.81
250	223.82	288.95	218.05	236.17
500	294.29	384.95	282.11	302.23
750	380.84	532.00	380.26	381.96
1000	626.70	683.95	580.99	521.49
1500	935.45	1073.13	808.98	620.17

Tabela 6.1: Custo de adaptação

O tempo total é composto por duas parcelas: o tempo de execução do adaptador, e o tempo gasto pelo proxy para tratar a mensagem, coletar e avaliar informações de contexto dos clientes e ativar a adaptação adequada. Este tempo é doravante chamado de *sobrecarga*.

Na Figura 6.2 estes dois componentes do tempo total são mostrados para a adaptação *Crop Center*, que realiza um recorte de tamanho 100x150 pixels no centro da imagem. Esta adaptação foi escolhida para ser utilizada na execução dos demais testes por apresentar um menor aumento de tempo de execução com aumento do tamanho da imagem.

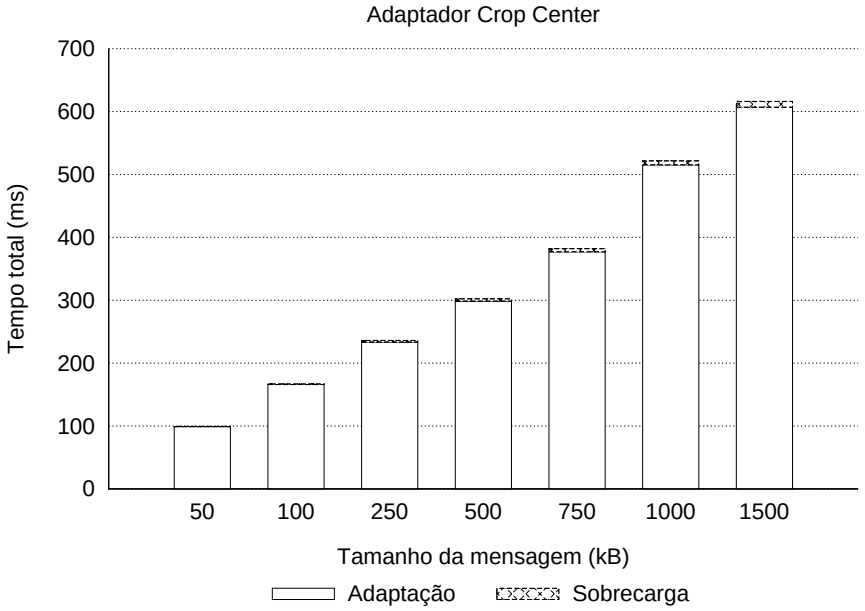

Figura 6.2: Tempo de sobrecarga e de adaptação para Crop Center

Pode-se notar que o tamanho da imagem influencia o tempo de adaptação, sendo o crescimento deste tempo quase linear com o aumento do tamanho da imagem. Além disso, apesar de bastante reduzida, a sobrecarga também sofre um aumento com o tamanho da mensagem. Por exemplo, a sobrecarga

é inferior a 1ms para imagens de 50kB, mas é de aproximadamente 10ms para imagens de 1500kB. Isso deve-se provavelmente ao maior tempo para manipulação da imagem na memória.

6.1.2 Custo Algoritmo Força Bruta versus Algoritmo com grupos

O objetivo deste experimento é comparar o desempenho dos algoritmos utilizados no gerenciamento e execução de adaptações, ou seja, o algoritmo força bruta (Algoritmo 1) e o algoritmo com grupos (Algoritmo 2).

As principais métricas de desempenho analisadas são: o tempo de serviço e o número de usuários atendidos pelo sistema. O tempo total de serviço representa o tempo gasto pelo proxy para aplicar as adaptações necessárias para adequar uma mensagem à condição de contexto corrente do cliente. Este tempo engloba tanto o gerenciamento dos clientes e seus grupos (tempo da sobregarca) quanto o processamento das adaptações (tempo de adaptação). Em relação ao número de usuários, deseja-se que o maior número possível de clientes possa ser atendido de forma satisfatória (isto é, mantendo-se um tempo de serviço razoável), ou seja, deseja-se mostrar que o sistema é escalável.

Neste experimento, variou-se o número de clientes (k) no intervalo [10, 1000]. Optou-se por utilizar três tamanhos de mensagens, a saber, pequeno (50 kB), médio (250kB) e grande (500kB). Além disso, os testes foram realizados com uma taxa de envio de duas mensagens por minuto. Esta taxa de envio é suficiente para boa parte dos serviços de disseminação de informações, como RSS por exemplo. Os resultados são apresentados confrontando o tempo de serviço médio com número de clientes. Os valores apresentados aqui, bem como em todos os demais experimentos, possuem intervalo de confiança de 95%, relativo a 20 execuções.

Inicialmente, foi avaliada o caso em que há apenas uma situação de contexto de interesse para a aplicação ($n = 1$) e apenas um adaptador(*Crop Center*) é acionado na ocorrência deste evento ($|A_{S1}| = 1$). Além disso, todos os clientes apresentam um estado ativo para a situação de contexto configurada, e portanto, a mesma adaptação é executada para todos os clientes.

O caso avaliado representa o melhor caso para o algoritmo com grupos, que possui um custo teórico de $O(d) = O(1)$. O algoritmo força bruta, por sua vez, tem um custo teórico de $O(n \times k) = O(k)$.

Os resultados do sistema que utiliza o algoritmo força bruta são apresentados nas Figuras 6.3 e 6.4. O primeiro gráfico apresenta o tempo médio de serviço, composto pela sobrecarga e pela adaptação, para diferentes tamanhos de mensagens e número de clientes. Pode-se notar que o tempo de adaptação

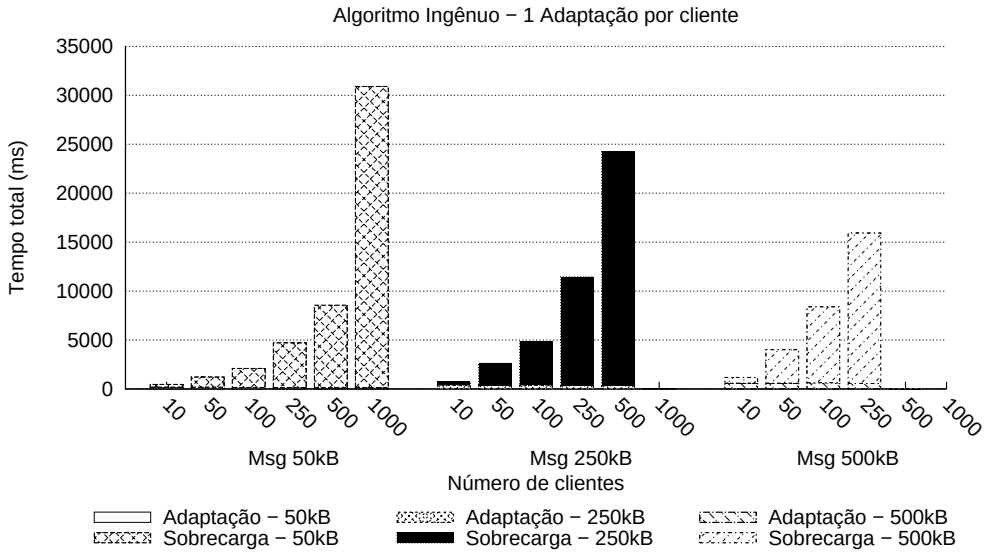

Figura 6.3: Tempo de serviço médio para o algoritmo força bruta, aplicando 1 adaptação

pouco se altera conforme o número de clientes aumenta, sendo quase não visível na base do histograma. Entretanto, o tempo médio de serviço aumenta de forma exponencial, conforme o número de clientes aumenta, principalmente devido à sobrecarga. A variação da sobrecarga é mostrada em detalhes no gráfico da Figura 6.4.

Para as mensagens maiores, de 250kB e 500kB, não foi possível finalizar os testes para 1000 e 500 clientes, respectivamente. Isto se deve à limitação de memória (512MB) e ao fato do algoritmo força bruta replicar a mensagem para cada cliente, gerando assim um grande número de cópias da imagem sendo adaptada. Como nos experimentos a freqüência de envio foi de 1 mensagem a cada 30 segundos e o tempo médio de serviço ultrapassava este valor, a taxa de criação de novas mensagens (réplicas) é superior a taxa de remoção (envio). A cada mensagem são criadas, por exemplo, 1000 cópias de 250kB, portanto, mais de 500MB em memória são alocados para mensagens logo no primeiro minuto, e este consumo de memória naturalmente aumenta durante a simulação.

O gráfico 6.4 apresenta a sobrecarga mínima, média e máxima para uma mensagem de tamanho 50kB. Nestes testes, sobrecarga significa todo o tempo gasto desde o momento da chegada da mensagem ao proxy até esta estar pronta para o envio ao cliente, subtraindo o tempo gasto nas adaptações. Ou seja, a sobrecarga inclui somente a gerência de clientes e de seus estados relativos às situações de contexto, e seleção de adaptações, e tempo de espera até o

adaptador poder ser executado, mas não a adaptação em si. Desta forma, tentamos isolar o custo do gerenciamento de clientes do tempo total de serviço, e assim verificar o quanto este custo cresce com o aumento do número de clientes nos dois algoritmos.

Pode-se notar a grande variação no tempo de atendimento dos clientes. Os primeiros são atendidos em um tempo (sobrecarga mínima) que varia de apenas 35ms, quando há 10 clientes, a 14,5s para 1000 clientes. A sobrecarga para os últimos clientes também aumenta significativamente a medida que aumenta-se o número de clientes atendidos. Por exemplo, para 10 clientes o máximo de espera é de 646 ms, enquanto que para 1000 clientes este tempo atinge 44,3s. Esta variação se deve principalmente ao tempo de replicação de mensagens, e ao tempo de espera (em filas) para que cada mensagem replicada possa ser adaptada. No caso de 1000 clientes pode-se notar o custo associado à replicação de mensagens até mesmo para os clientes atendidos primeiro, que esperam no mínimo 14,5 segundos. Provavelmente, isso se deve à necessidade de paginação de memória.

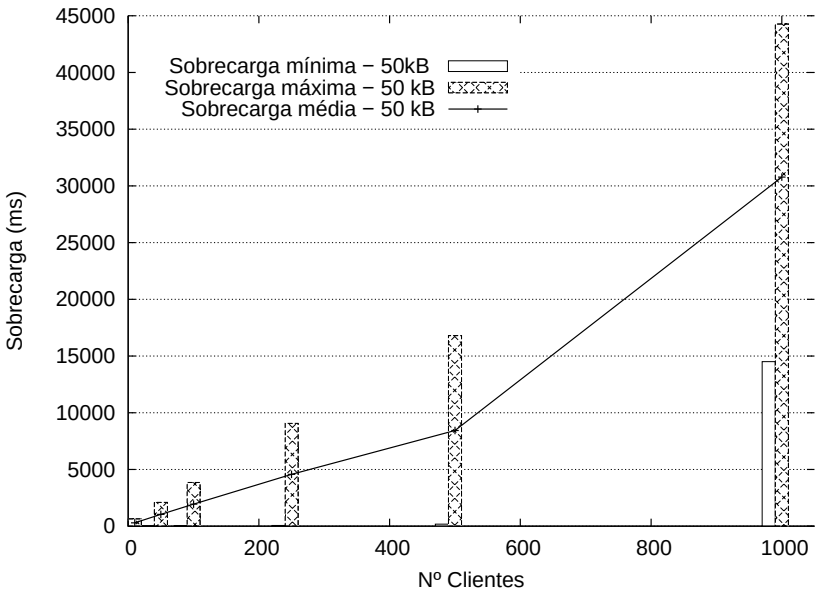

Figura 6.4: Sobrecarga do algoritmo força bruta enviando mensagem de 50kB

A Figura 6.5 apresenta um gráfico similar ao da Figura 6.3, mas agora usando-se o algoritmo com grupos e considerando os mesmos parâmetros de teste do algoritmo força bruta mostrado anteriormente. Esta figura apresenta o tempo de serviço total, composto pelo tempo de adaptação e pela sobrecarga, para diferentes tamanhos de imagens. Como todos os clientes executam a mesma adaptação, apenas 1 grupo é criado, e a adaptação é executada uma única vez. Isso caracteriza o melhor caso do algoritmo com grupos. Pode-se

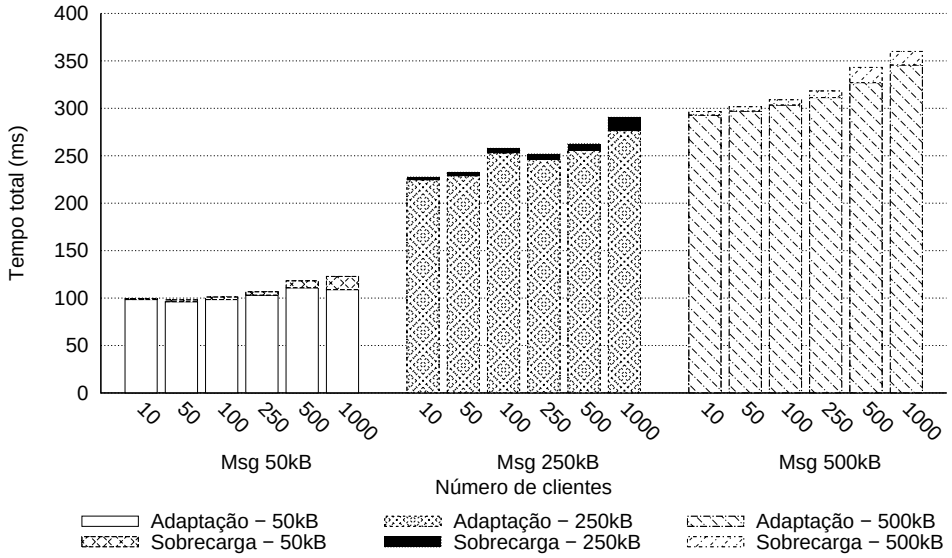

Figura 6.5: Tempo serviço para algoritmo com grupos

notar que o tempo da adaptação é muito próximo ao apresentado na Figura 6.2, com um ligeiro aumento conforme o número de clientes aumenta. A sobrecarga, como esperado, tem um pequeno aumento quando o número de clientes aumenta, já que é necessário avaliar o estado de contexto de cada cliente. No entanto, essencialmente, o tempo de serviço é dominado pelo tempo de adaptação, ficando ligeiramente acima deste devido à sobrecarga de gerenciamento de clientes, que neste caso é pequeno (inferior a 20ms para 1000 clientes), pois, neste caso, todos os clientes estão em apenas um grupo.

Tempo de serviço total médio

Comparando o algoritmo força bruta (Figuras 6.3 e 6.4) com o algoritmo com grupos (Figura 6.5) pode-se notar que este melhor caso do algoritmo com grupos apresenta um desempenho médio de aproximadamente 5 a 250 vezes superior ao algoritmo força bruta, de acordo com o tamanho da mensagem e número de clientes. Por exemplo, o tempo médio para enviar mensagens de tamanho 250kB a 500 clientes no algoritmo força bruta é de 25 segundos, enquanto que no algoritmo com grupos este tempo cai para 260ms, ou seja, um custo quase 100 vezes menor.

É interessante notar que o algoritmo com grupos, além de reduzir o tempo de sobrecarga, que mais influencia no tempo de serviço do algoritmo força bruta, também executa as adaptações em um tempo inferior. Enquanto que para o algoritmo com grupos a adaptação demora em média apenas 5% a mais do que a adaptação pura (medido no teste da Figura 6.2), para o algoritmo

força bruta as adaptações demoram em média de 55% a 95% a mais. Este aumento no tempo de execução da adaptação CropCenter no algoritmo força bruta deve-se provavelmente ao fato de haver maior custo para controle de concorrência na execução das adaptações (pelas threads), devido ao grande número de mensagens replicadas.

Faixa de variação do tempo médio de serviço

Outro ponto a se notar é a maior variação do tempo médio de serviço do algoritmo força bruta. No algoritmo com grupos há apenas uma pequena variação no tempo de serviço com o aumento do número de clientes, devido ao pequeno aumento da sobrecarga com a gerência de clientes. Isto ocorre porque, neste experimento (Figura 6.5), há a formação de apenas um único grupo para a adaptação das mensagens, sendo o tempo de médio de serviço igual para todos os clientes. Como visto anteriormente, no algoritmo força bruta, o tempo de serviço é dominado pelo tempo de sobrecarga. Analisando a Figura 6.4, pode-se notar que neste algoritmo o tempo de serviço observado pelos clientes (para mensagens de 50 kB) pode variar de 0,5s a 45s, ou seja, uma variação de até 90 vezes. Esta variação tende a piorar tanto para um maior número de clientes, quanto para mensagens de maior tamanho.

6.1.3 Algoritmo com grupos para 5 regras de adaptação

Neste experimento, o algoritmo com grupos foi executado considerando cinco situações de contexto de interesse ($n = 5$) para adaptações. O objetivo é determinar a influência de um número maior de regras no tempo de resposta do sistema. Dois casos foram avaliados, o primeiro para verificar a sobrecarga pura da gerência de grupos, e o segundo para comparar o efeito do aumento do número de situações de contexto em relação ao cenário da seção anterior (1 situação de contexto - Figura 6.5).

Os testes a seguir consideram o cenário hipotético no qual para cada uma das cinco situações de contexto é executada uma adaptação. Para uniformizar o custo das adaptações, utilizou-se o mesmo adaptador (*Crop Center*), mas com parâmetros ligeiramente diferentes, para cada situação. Assim, o custo de adaptação das diferentes situações de contexto é bastante próximo, para não distorcer a média do tempo de serviço. O intervalo de confiança permanece em 95%.

Para verificar o custo operacional (gerência de clientes e seus contextos, e encaminhamento de mensagens) do sistema com algoritmo de grupos, executou-se um teste no qual nenhum cliente encontrava-se com estado de con-

texto igual a nenhuma das cinco situações. Desta forma, todos os clientes ficam em um mesmo grupo, mas as mensagens não sofrem qualquer adaptação. Isto configura o caso mais favorável, fazendo com que os resultados apresentados na Figura 6.6 apresentem a sobrecarga pura do sistema, ou seja, aquela causada pela gerência de contextos de clientes recebidos do Serviço de Contexto, e avaliação das regras de adaptação.

Pode-se notar que, por não haver formação de novos grupos, e não haver espera nas filas de atendimento, o aumento do número de clientes leva a um aumento de até 25ms no tempo de sobrecarga, para 1000 clientes. O tamanho da mensagem também contribui para o aumento na sobrecarga. Para mensagens pequenas (50kB) a sobrecarga varia de 0,5 a 23ms, enquanto que para mensagens grandes (500kB) este valor varia de 15 a 28ms, para 10 a 1000 clientes. Note que para imagens maiores e poucos clientes a sobrecarga média inicial é também maior. Esta média deve estar sendo influenciada pelo tempo de carga inicial de objetos que contêm a mensagem, e como são poucos clientes, este tempo não é diluído na média.

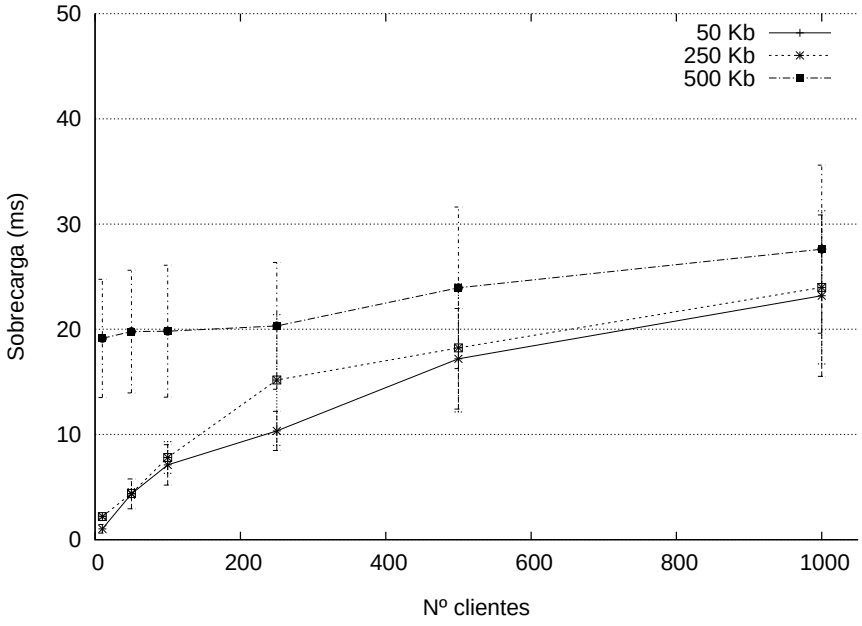

Figura 6.6: Overhead puro para algoritmo com grupos (sem adaptação)

As figuras 6.7 e 6.8 apresentam os resultados para o algoritmo com grupos no cenário com 5 situações de contexto e onde cada cliente enquadra-se em apenas uma das situações de contexto de interesse, e apenas 1 adaptação é executada por mensagem. Os simuladores do monitor (Monitor Simulator) foram configurados de forma a enviar os dados de contexto de um cliente ao CIS a cada 1 segundo. Os eventos de mudança do contexto (dos clientes) correspondentes foram escolhidos de tal forma que os clientes ficassem uniformemente

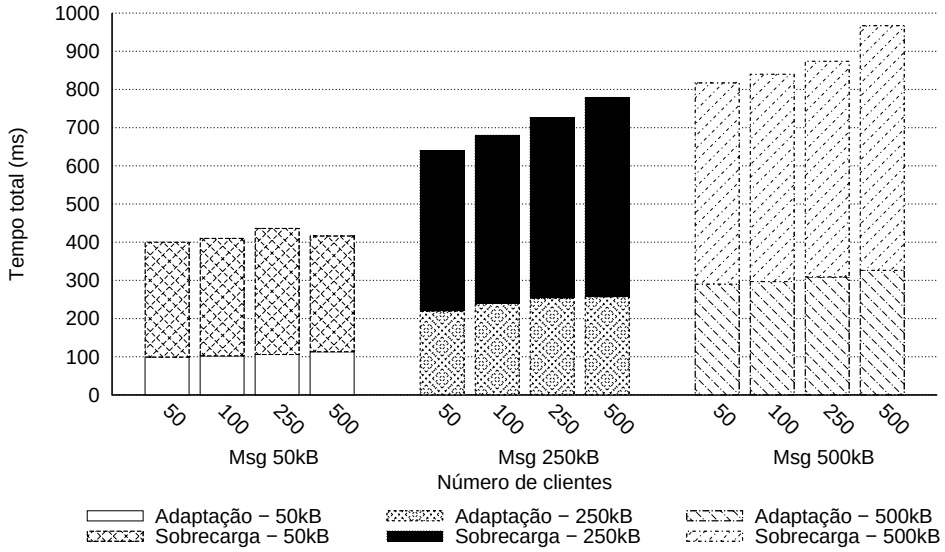

Figura 6.7: Tempo de Serviço: 5 situações 20% de clientes por estado 1 adaptação

distribuídos em cada grupo, isto é, para cada situação de contexto havia aproximadamente 20% de clientes ativos.

A Figura 6.7 apresenta o tempo médio de serviço total, e suas parcelas de adaptação e de sobrecarga para diferentes tamanhos de mensagens e número de clientes. Pode-se notar que o tempo de adaptação continua similar ao da Figura 6.2, e pouco se altera com o aumento do número de clientes. Entretanto, pode-se notar que, ao contrário do cenário com apenas uma situação de contexto (Figura 6.5), a sobrecarga representa o maior custo, e é o que mais determina o tempo total, e aumenta proporcionalmente ao tamanho da mensagem transmitida e ao número de clientes. Por exemplo, para uma mensagem de tamanho 250kB e 500 clientes, o tempo médio de serviço é aproximadamente 800ms, enquanto que quando havia apenas uma situação de contexto, este tempo era inferior a 300ms. Como a adaptação possui um tempo similar em ambos os casos, a diferença de tempo é devido à sobrecarga, que passou de 50ms para 500ms. Este aumento da sobrecarga deve-se, principalmente, ao tempo de espera para a execução de cada adaptação. Apesar de haver um pool de threads para executar as adaptações para cada grupo, essas adaptações consomem muitos ciclos de CPU.

Outra diferença deste cenário em relação ao cenário com uma única situação de contexto (Figura 6.5) é a variação do tempo de serviço. No cenário da seção anterior (Figura 6.5), o tempo de serviço era igual para todos os clientes, pois havia apenas 1 situação de contexto, e portanto, formava-se apenas um grupo para adaptação. Neste experimento, como há a formação

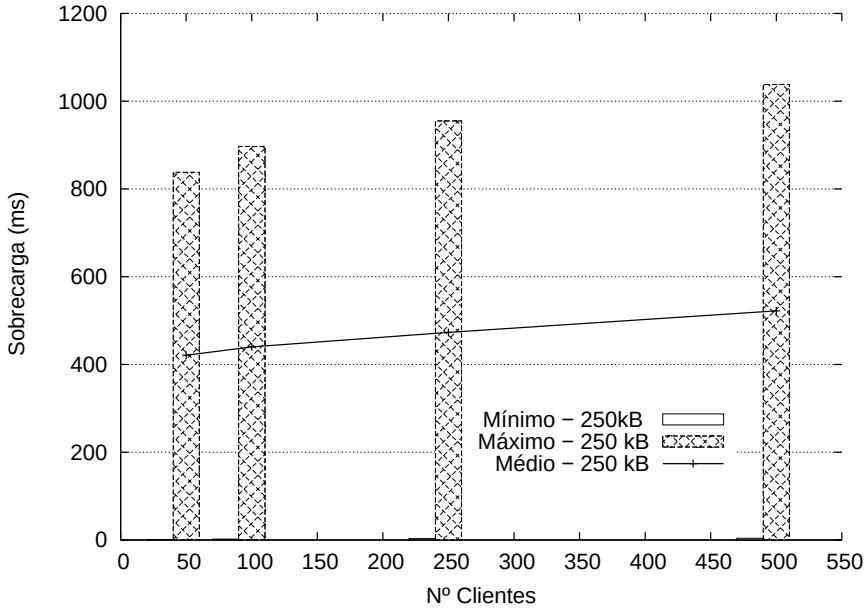

Figura 6.8: Sobrecarga mínima, média e máxima para mensagem de 250kB, em sistema com 5 situações de contexto

de 5 grupos distintos, o tempo de serviço mostrado é a média dos tempos de serviço dos grupos. Como o tempo de adaptação é quase constante, a Figura 6.8 apresenta a variação do tempo de sobrecarga, que provoca a variação no tempo de serviço. São apresentados os tempos mínimo, médio e máximo de sobrecarga para diferentes números de clientes, para uma mensagem de tamanho 250kB. Em todas as situações, a sobrecarga mínima é próxima a zero (de 2 a 5 ms), enquanto que a sobrecarga máxima varia em torno de 15% com o aumento do número de clientes, atingindo mais de 1000ms para 500 clientes.

O experimento não foi realizado para o algoritmo força bruta, pois como nele a mensagem é replicada para todos os clientes, e como as adaptações para as 5 situações de contexto são semelhantes, não há diferença significativa em relação aos resultados apresentados nas Figuras 6.3 e 6.4. O tempo de serviço do algoritmo com grupos, mesmo tendo piorado em relação ao cenário com 1 situação de contexto de interesse, ainda apresenta um resultado até 50 vezes melhor que o algoritmo força bruta.

Montagem do cenário do experimento com 5 situações de interesse

Como já mencionado, neste experimento foram configuradas 5 situações de interesse no ProxyFramework. Cada situação de contexto demanda apenas uma adaptação, e um grande número de clientes simulados foram uniformemente distribuídos nestas situações. A seguir é explicada a forma como os Monitor Simulators foram configurados para se montar o cenário deste experi-

mento, e o motivo pelo qual neste experimento o número de clientes simulados ficou restrito a 500.

O Monitor Simulator lê as informações de contexto relativas a vários dispositivos clientes de um arquivo com uma estrutura própria, e as envia ao CIS de acordo com a periodicidade definida. As informações de contexto para cada dispositivo cliente são definidas em uma linha do arquivo. A cada intervalo de tempo uma linha é lida e enviada ao CIS, e ao término do arquivo, o simulador recomeça sua leitura do início do arquivo.

Nos experimentos, o intervalo de envio usado foi de 1 segundo. Ou seja, a cada 1 segundo informações sobre um dispositivo cliente eram transmitidas ao CIS. Portanto, se as informações de contexto de todos os 1000 dispositivos clientes estivessem em um mesmo arquivo, seriam necessários aproximadamente 17 minutos para que o CIS recebesse as informações de todos os clientes. Na implementação atual do CIS, as condições de contexto de um cliente são avaliadas apenas no momento em que os dados do monitor são recebidos pelo CIS, e portanto seriam necessários 17 minutos para notificar o proxy sobre as condições de todos os clientes, o que tornaria a execução dos testes em lote praticamente impossível.

Por esse motivo, foi necessário separar as informações de contexto dos dispositivos clientes simulados em arquivos representando 20 clientes cada, para que a informação de um cliente demorasse no máximo 20s para estar disponível ao proxy. Como cada arquivo é lido por um Monitor Simulator diferente, aumentar o número de clientes, bem como variar as condições de contexto destes, mostrou-se uma tarefa bastante trabalhosa.

Além disso, outro fator que limitou o número de clientes simulados foi a sobrecarga de processamento percebida no CIS. Por exemplo, para 500 clientes, foram usados 25 Monitor Simulators (20 clientes por simulador), portanto o CIS recebia 25 pacotes de informações de contexto por segundo (um pacote de cada monitor). Neste intervalo de 1s o CIS deve processar as informações dos 25 clientes para determinar se os clientes encontram-se em uma ou mais situações de contexto solicitadas, e em caso positivo, notificar o proxy. Aumentar o número de clientes implica em aumentar esta sobrecarga, o que pode causar perda de informações, ou impossibilidade de geração de eventos na freqüência esperada. Isto poderia invalidar os testes, devido a falta de garantia da geração e manutenção do cenário de teste planejado.

6.1.4 Variação do número de adaptações por cliente para Algoritmo com Grupos

Apesar dos experimentos descritos nas seções anteriores utilizarem simuladores para gerar as informações de contexto dos dispositivos clientes e enviá-las ao serviço de contexto (CIS), os experimentos são próximos à realidade, pois incluem a interação real do proxy com CIS, e estão sujeitos a eventuais problemas, como perda de informações causados por atrasos na comunicação ou no envio de eventos.

Por outro lado, devido às dificuldades de configuração dos cenários de teste (descritos na seção anterior), tornou-se inviável a realização testes de desempenho com um maior número de clientes. Além disso, a forma de configuração dos Monitor Simulators torna muito trabalhosa a criação de cenários nos quais há variação do número de adaptações efetuadas para cada cliente, e variação da quantidade de clientes executando diferentes subconjuntos de adaptadores.

Por estes motivos, os experimentos apresentados nas próximas seções utilizaram um simulador de estados de contextos dos clientes, de forma a tornar tais estados mais determinísticos. Os resultados do algoritmo com grupos são também confrontados com os resultados do algoritmos força bruta e ingênuo submetidos às mesmas condições.

Descrição e parâmetros dos experimentos com simulador de estados de contexto

Os experimentos a seguir têm a finalidade de avaliar o impacto da variação do número de adaptações por cliente no tempo médio de serviço. Os testes consideram o cenário com cinco situações de contexto, cada uma demandando uma única adaptação. Usou-se como número de clientes os valores: 250, 500 e 1000, e como número de possíveis situações de contexto assumidos por cada cliente entre 0 à 5, ou seja, usou-se uma variação de 0 a 5 adaptações por cliente. Como nos experimentos anteriores, utilizou-se para todas as situações o mesmo adaptador (*CropCenter*), mas com parâmetros ligeiramente diferentes. Mensagens pequenas, com imagem de tamanho 50kB, foram adotadas para os demais testes, pois este tamanho reflete um tamanho de arquivo comum a boa parte das imagens transmitidas via WEB.

Como dito anteriormente, foi necessário simular os estados de contexto dos clientes para realizar testes mais complexos, nos quais os clientes pudessem estar com várias situações de contexto ativas em cada momento.

Para esta simulação, foi utilizado um algoritmo que realiza a distribuição dos clientes nos grupos formados durante a execução da recursão do algoritmo

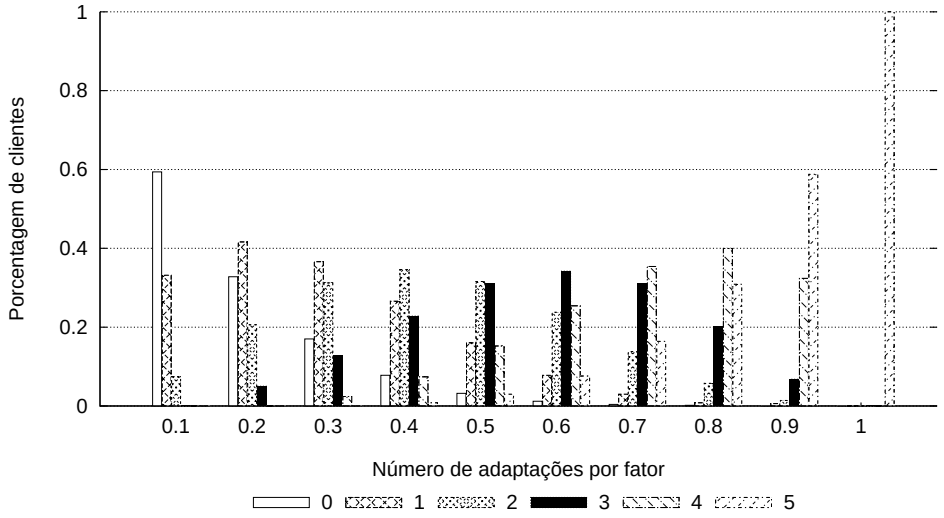

Figura 6.9: Distribuição de clientes que necessitam de 0 a 5 adaptações de acordo com o fator de quebra de grupo.

com grupos (Algoritmo 2). De acordo com a profundidade da árvore de recursão (situação de contexto i), são criados 2^i grupos, sendo a metade destes grupos compostos de clientes com estado ativo (G_{Si}). A proporção de clientes ativos para cada situação de contexto i é determinada pelo fator de quebra do grupo (g - definido na Seção 4.3.3) adotado na simulação. Assim, para cada situação de contexto é selecionada uma quantidade de clientes ativos igual ao fator de quebra multiplicado pelo número total de clientes ($g \times k$). Estes clientes são então distribuídos uniformemente pelos grupos de ativos (G_{Si}). As situações de contexto e o fator de quebra dos grupos são definidos no início da simulação, e não sofrem alteração durante a execução da mesma, para tornar o comportamento determinístico e possível de ser reproduzido em diferentes simulações.

A Figura 6.9 mostra o resultado do algoritmo de simulação dos estados de contexto para 500 clientes. É apresentada a proporção de clientes que necessitam de 0 a 5 adaptações, em função do fator de quebra de grupo. A distribuição para 250 e 1000 clientes é praticamente a mesma em termos percentuais, e por isso foi omitida. Note que o fator de quebra determina a quantidade de adaptações selecionadas para os clientes. Para fatores g baixos, quase inexistem casos em que clientes necessitam de mais de 2 adaptações. Conforme o fator de quebra aumenta, aumenta também a proporção de clientes que requerem um maior número de adaptações.

Os demais experimentos adotam esta distribuição de clientes em relação ao número de adaptações efetuadas, de acordo com a variação do fator de

quebra.

Resultados

Neste experimento, o objetivo é avaliar o impacto do aumento do número de adaptações realizadas por cliente no tempo de serviço. Mas, como a variação do número de adaptações realizadas por cliente depende do fator de quebra de grupo escolhido para a simulação, os gráficos são apresentados em função deste fator.

A Figura 6.10 exibe o tempo médio de serviço para o algoritmo com grupos, para 250, 500 e 1000 clientes, em função da variação do fator de quebra (ou proporção de clientes que precisam de adaptação em cada situação de contexto). O tempo médio de serviço, neste gráfico, é uma média ponderada pelo número de clientes em cada grupo. Desta forma grupos com maior número de clientes têm maior influência sobre o resultado. Com isso pretende-se verificar como o tempo de serviço é percebido pela maioria dos clientes.

Para auxiliar a avaliação do gráfico de tempo médio de serviço, a Figura 6.11 apresenta o número de grupos criados ao final da simulação, em função do fator de quebra. Pode-se notar que o número de grupos varia de no mínimo 1, no melhor caso, a no máximo 32, devido ao número de situações de contexto simuladas ser igual a 5. Ou seja, número máximo de grupos é igual a potência de dois do número de situações de contexto (2^n).

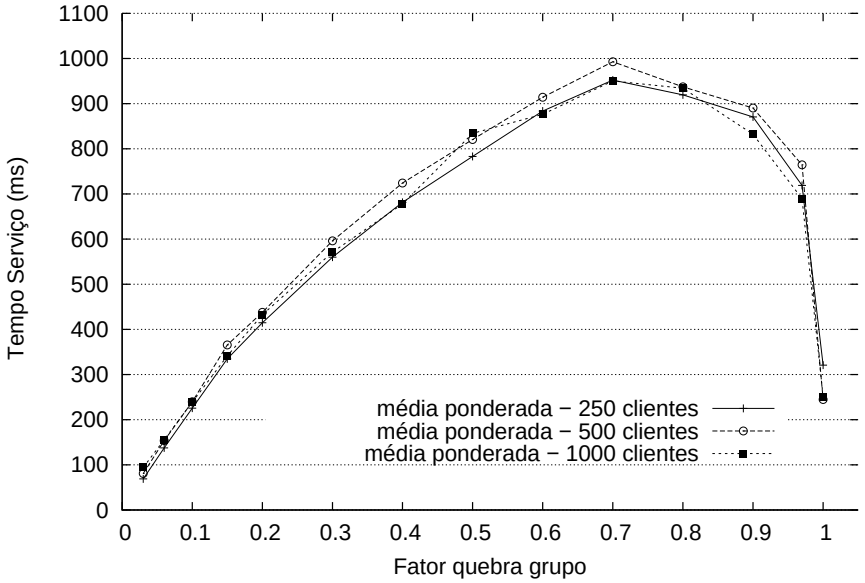

Figura 6.10: Tempo médio de serviço de acordo com o fator de quebra.

Verificando-se a Figura 6.10 em conjunto com as figuras de grupos (6.11) e de distribuição de clientes (6.9), pode-se notar que, para fatores de

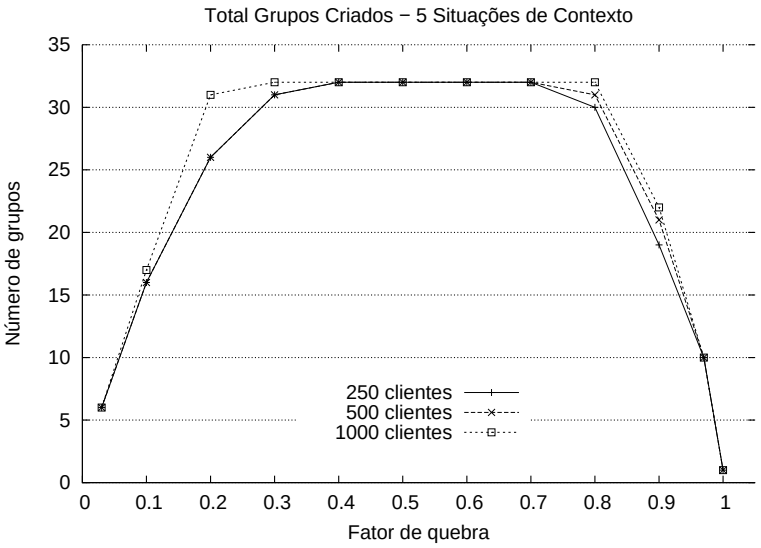

Figura 6.11: Grupos criados em função do fator de quebra.

quebra menores, o tempo de médio serviço mantém-se baixo, pois os clientes concentram-se nos grupos que executam poucas adaptações. Conforme o fator de quebra aumenta, aumenta também o número de clientes que efetuam um número maior de adaptações (Figura 6.9), e portanto o tempo médio de serviço também aumenta. Por outro lado, na Figura 6.11 verifica-se que o número de grupos atinge seu valor máximo em torno de um fator de quebra de 0,4 (devido a elevada quantidade de clientes, e da distribuição uniforme destes entre os grupos) e volta a cair a partir do fator de quebra 0,7. Quanto maior for o número de grupos, maior é o tempo de serviço, já que um número maior de adaptações repetidas são realizadas. Além disso, devido à forma como o algoritmo é executado (recursão sobre o número de situações de contexto), ocorre que os clientes com necessidade de 0 adaptações são os primeiros a serem atendidos, e os com 5 adaptações são os últimos.

O pior caso, para este experimento (considerando a média ponderada), fica em torno do fator 0.7, quando o tempo médio de serviço experimentado pela maioria dos clientes atinge 1s. Isso decorre do fato do número de grupos ainda ser máximo, com a maioria deles necessitando de 3 ou mais adaptações. A partir do fator 0.7, apesar da maior parte dos clientes necessitar de 4 ou 5 adaptações, estes clientes estão mais agrupados, e portanto o tempo médio de serviço começa novamente a decrescer. Uma queda mais acentuada no tempo de serviço é percebida após o fator de quebra 0.95, devido à redução no número de grupos. Finalmente, para fator de quebra igual a 1, isto é, quando todos os clientes necessitam das 5 adaptações, forma-se apenas um grupo e portanto cada uma das 5 adaptações é executada apenas uma única vez. Assim, a

média ponderada do tempo de serviço é aproximadamente 230ms para todos os clientes.

Outro aspecto a ser observado é que o aumento do número de clientes não causou impacto sobre o tempo de serviço, mostrando assim a escalabilidade do sistema. Isso se deve ao fato de que o aumento de clientes apenas deixa os grupos com um número maior de clientes, mas o número de adaptações total permanece o mesmo.

A Figura 6.12 detalha o tempo médio de serviço para 500 clientes, apresentando suas duas componentes: tempo de execução das adaptações e tempo de sobrecarga, sendo que esta última engloba o tempo de gerência de grupos do algoritmo e o tempo de espera até o processamento da adaptação. Neste gráfico, o tempo de serviço é mostrado em relação ao número de adaptações realizadas, e representa uma média simples dos tempos experimentados por todos os clientes que necessitaram de um certo número de adaptações. Como esperado, o tempo de serviço aumenta significativamente com o aumento do número de adaptações realizadas, sendo influenciado principalmente pelo tempo de sobrecarga. Entretanto, neste experimento, a parcela de tempo relativa à adaptação não aumenta linearmente com o aumento do número de adaptações realizadas, como poder-se-ía esperar. Isso se deve ao tipo de adaptador utilizado no experimento, que reduz o tamanho da imagem. Portanto, o tempo gasto para realizar a segunda adaptação é bem inferior ao da primeira, o tempo da terceira é inferior ao da segunda adaptação, e assim sucessivamente. Isso pode não acontecer com outros tipos de adaptadores, como os de conversão de formatos, mas é uma situação bastante comum, visto que boa parte das adaptações para clientes móveis resultam em algum tipo de perda de informação e, conseqüentemente, em redução do tamanho da mensagem transferida.

O tempo médio de serviço dos clientes que não realizam nenhuma adaptação é próximo a zero, pois são os primeiros a serem atendidos, tendo assim, uma sobrecarga mínima, além de não não ser necessário realizar adaptações. A sobrecarga aumenta de forma sub-linear com o aumento do número de adaptações, e também sofre variações em relação ao fator de quebra, crescendo com o número de grupos, até se estabilizar em torno de 0.4 a 0.7, quando volta a ter uma queda. Esta variação é pequena pois neste gráfico apresenta-se uma média simples, e os clientes estão uniformemente distribuídos pelos grupos. Portanto, o tempo de espera para a adaptação é similar para os diferentes fatores de quebra, variando apenas a gerência de grupos. No pior caso, a sobrecarga atinge aproximadamente seis vezes o tempo de adaptação (vide fator 70%).

Ao se comparar o tempo de sobrecarga para os clientes que efetuaram

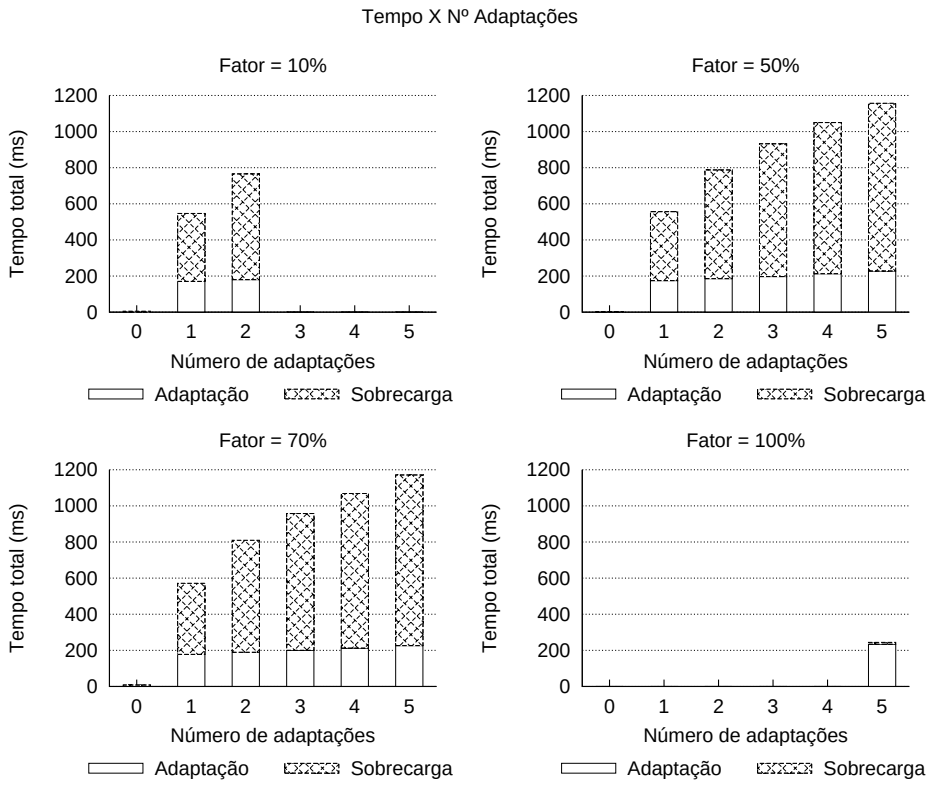

Figura 6.12: Tempo médio de serviço por número de adaptações - 500 clientes

apenas uma adaptação com o resultado obtido na Figura 6.7 (item mensagem de tamanho 50kB e 500 clientes), no qual havia as mesmas 5 situações de contexto, mas cada cliente efetuava apenas 1 adaptação, verifica-se que houve um aumento do tempo de sobrecarga de 300ms (Figura 6.7) para 370ms (Figura 6.12). Este aumento se deve ao maior número de grupos criados no experimento corrente, o que também aumenta o tempo de espera nas filas, pois há clientes que executam apenas uma adaptação mas que precisam esperar o processamento de outros grupos com 2 ou mais adaptações, devido à ordem de execução (percorrimento da árvore de grupos em-ordem).

Comparando a Figura 6.12 de fator 100% com a Figura 6.5, na qual todos os clientes executavam uma única adaptação, o tempo médio de serviço aumentou de 120ms para 240ms. Note que, neste experimento, são realizadas 5 adaptações, e não apenas 1 como no caso da figura 6.5, e este aumento do tempo de serviço deve-se exclusivamente ao maior número de adaptações, já que a sobrecarga variou apenas de 7ms para 10ms.

6.1.5 Variação do número de adaptações para o Algoritmo Força Bruta

Os mesmos parâmetros do experimento anterior foram utilizados para executar testes do proxy instanciado com o algoritmo força bruta de gerência de adaptações, a fim de comparar seus resultados com os resultados do algoritmo com grupos.

O tempo médio de serviço do algoritmo força bruta é apresentado na Figura 6.13, em função do fator de quebra, que neste caso significa porcentagem de clientes que necessitam de adaptação em cada situação de contexto. Para o algoritmo força bruta, o aumento do número de clientes impacta significativamente no desempenho do proxy.

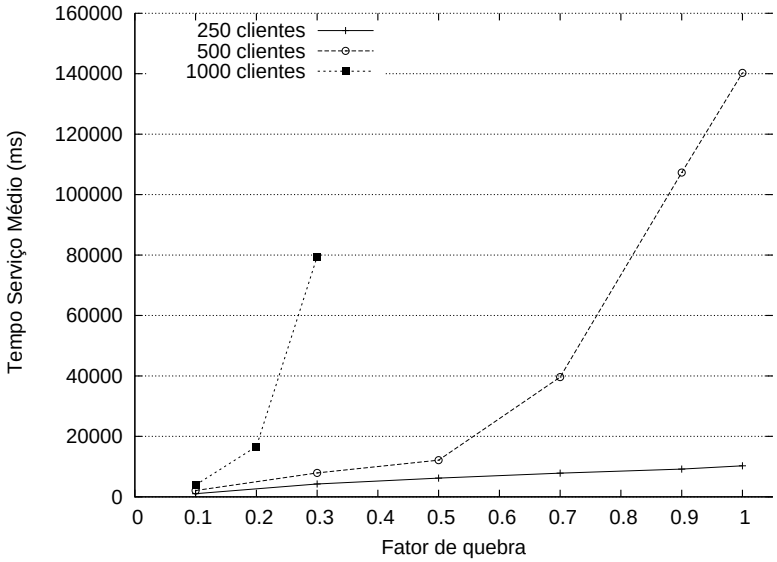

Figura 6.13: Tempo médio de serviço do algoritmo força bruta

Como visto na Figura 6.10, o aumento do número de clientes pouco influencia o tempo médio de serviço no algoritmo com grupos. Entretanto, para o algoritmo força bruta, o maior número de adaptações implica em um aumento exponencial do tempo médio de serviço. Além disso, a porcentagem de clientes que necessitam de adaptações também causa um grande impacto no tempo médio de serviço do algoritmo força bruta. Isso se deve ao fato de que nenhuma adaptação é compartilhada e todas as mensagens são replicadas. Enquanto que, para o fator de 100%, o algoritmo com grupos apresenta o resultado ótimo, com tempo de serviço médio de 230ms, para o algoritmo força bruta este é o pior caso, no qual o tempo de serviço atinge 1s para 250 clientes e 140s para 500 clientes, ou seja, neste caso (fator 100%) o algoritmo com grupos pode ser até 560 vezes melhor que o força bruta. Não foi possível

finalizar os testes para fatores acima de 0.3 com 1000 clientes devido à limitação de memória.

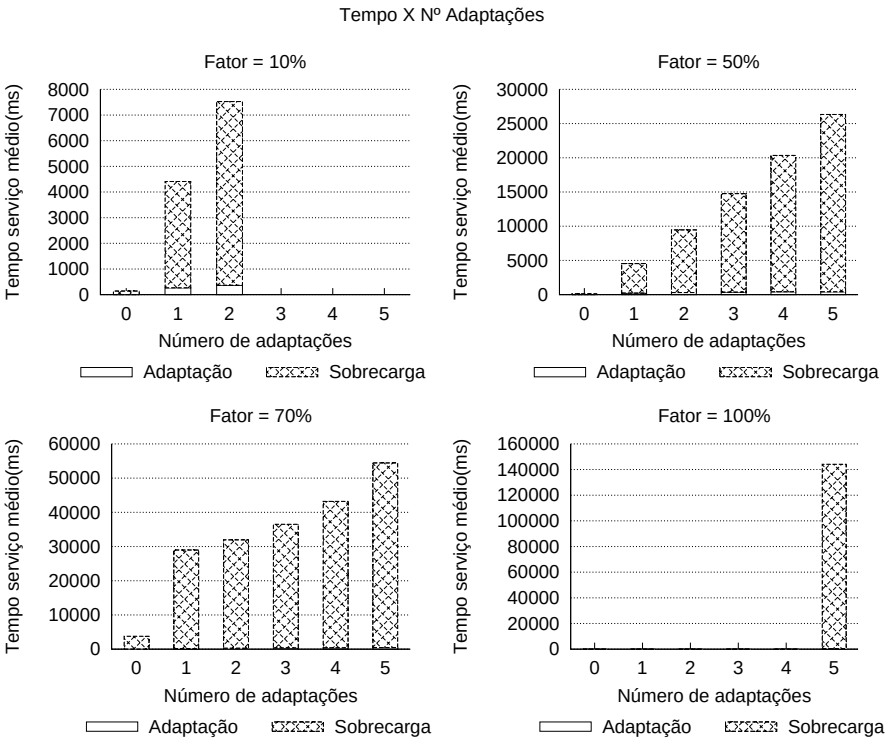

Figura 6.14: Tempo de serviço por número de adaptações - algoritmo força bruta - 500 clientes

Na Figura 6.14, são apresentadas as parcelas do tempo médio de serviço referentes à adaptação e à sobrecarga do algoritmo força bruta para 500 clientes, em função do número de adaptações efetuadas. O tempo de adaptação, mesmo para os clientes que necessitam de cinco adaptações, não é significativo quando comparado ao tempo médio de sobrecarga. O tempo de sobrecarga cresce linearmente com o aumento do número de adaptações (vide fator 50%). Além disso, o aumento da proporção de clientes com necessidade de adaptações também tem grande impacto sobre o tempo de sobrecarga, por exemplo, para cinco adaptações, o tempo aumenta de 25s (fator 50%) para 140s (fator 100%). Esta situação é bastante distinta daquela do algoritmo com grupos (Figura 6.12), no qual o aumento do número de adaptações aumenta o tempo de sobrecarga de maneira sub-linear, e o fator de quebra influencia o tempo de sobrecarga de forma aproximadamente parabólica. Pode-se perceber, que mesmo para proporções pequenas de clientes que necessitam de adaptações, por exemplo 10%, o algoritmo força bruta é aproximadamente 10 vezes mais lento que o algoritmo com grupos. Enquanto, para o algoritmo com grupos, o tempo médio de sobrecarga se estabiliza com valores entre 350ms e 900ms, de acordo com o número de adaptações (1 a 5), nos fatores de quebra 0.4 a 0.7, o

tempo de sobrecarga para o algoritmo força bruta sempre aumenta, sendo, por exemplo, até 80 vezes superior ao algoritmo com grupos (para fator de 70%), cujos tempos são de 30s a 55s dependendo do número de adaptações.

Entretanto, vale ressaltar que o fraco desempenho do algoritmo força bruta pode ser melhorado incorporando algumas alterações simples em seu funcionamento. Uma possível modificação é a separação dos clientes em classes. Cada classe de clientes é composta por clientes que possuem exatamente a mesma situação de contexto, ou seja, clientes cujas mensagens devem sofrer a mesma seqüência de adaptações. Esse algoritmo, doravante chamado *algoritmo ingênuo*, é utilizado nos experimentos da próxima seção para uma comparação mais justa com algoritmo proposto (Algoritmo 2) e para obter medidas mais precisas do ganho de desempenho e escalabilidade deste.

6.1.6 Variação do número de adaptações para o Algoritmo Ingênuo

Este experimento repete o cenário anterior com 5 situações de contexto, mas utiliza o algoritmo ingênuo para a gerência de adaptações, com o intuito de comparar seus resultados com os resultados do algoritmo com grupos. É avaliado qual o impacto no tempo de serviço em relaçao à variação do número de clientes e do número de adaptações realizadas por cada cliente.

O tempo médio de serviço do algoritmo ingênuo é apresentado na Figura 6.15, em função do fator de quebra, que representa a proporção de clientes que demandam adaptações em cada situação de contexto.

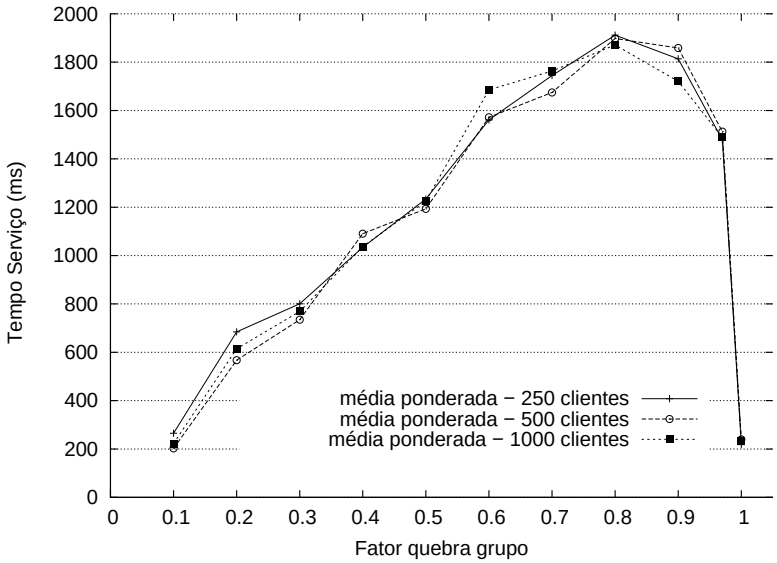

Figura 6.15: Tempo médio de serviço do algoritmo ingênuo

Pode-se notar que, ao contrário do algoritmo força bruta (Figura 6.13),

e analogamente ao algoritmo com grupos (Figura 6.10), o aumento do número de clientes pouco influencia o tempo médio de serviço do algoritmo ingênuo. Além disso, da mesma forma como no algoritmo com grupos, o tempo de serviço aumenta com o aumento do fator de quebra, devido a proporção de clientes que executam adaptações ser maior, mas a partir de um fator de aproximadamente 0.8, o tempo de serviço volta a cair, devido ao maior agrupamento dos usuários. Entretanto, neste algoritmo, a porcentagem de clientes que necessitam de adaptações causa um maior impacto no tempo médio de serviço do que no algoritmo com grupos. A diferença cresce com o aumento do fator de quebra, iniciando em 1% (fator= 0.1) e atingindo 110% em torno do fator de quebra igual a 0.8. Finalmente, para o fator 1, o desempenho dos algoritmos com grupos e do algoritmo ingênuo novamente se equiparam, com tempo de serviço médio pouco acima de 200ms, pois em ambos os algoritmos há a formação de apenas um grupo de clientes, todos necessitando das 5 adaptações.

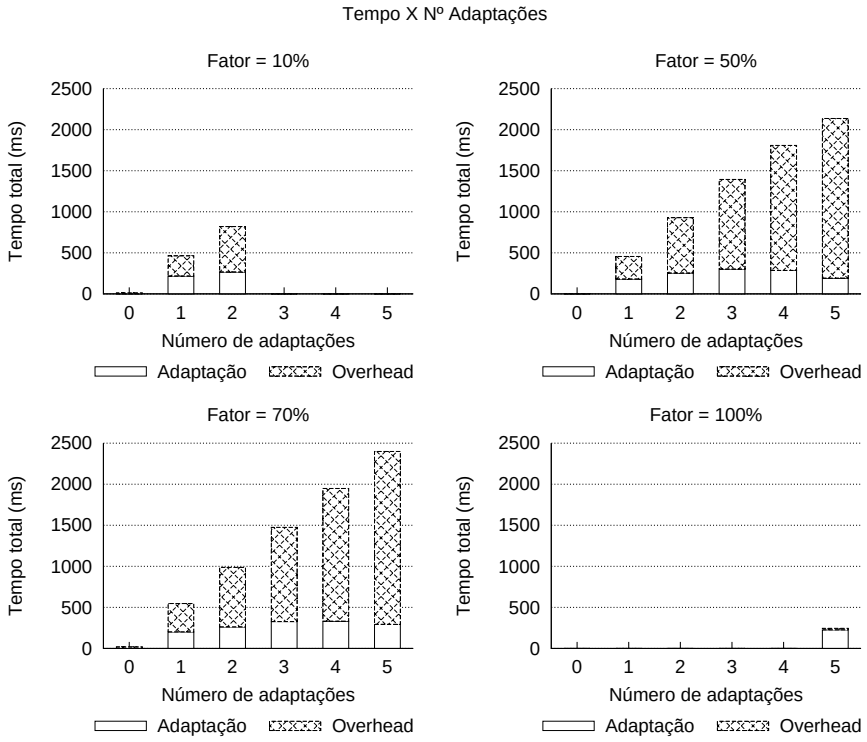

Figura 6.16: Tempo de serviço por número de adaptações - algoritmo ingênuo - 500 clientes

A Figura 6.16 apresenta as parcelas do tempo médio de serviço referentes à adaptação e à sobrecarga do algoritmo ingênuo para 500 clientes, em função do número de adaptações efetuadas. Da mesma forma como no algoritmo com grupos, o tempo de adaptação pouco aumenta entre os clientes que fazem de 1 a 5 adaptações, devido ao tipo de adaptação (com redução da imagem) utilizada nos testes. O aumento no tempo de serviço se deve ao aumento

do tempo de sobrecarga, que cresce quase linearmente com o aumento do número de adaptações (vide fator 70%). Além disso, o aumento da proporção de clientes com necessidade de adaptações também causa um certo impacto sobre o tempo de sobrecarga, por exemplo, para cinco adaptações, o tempo aumenta de 2s (fator 50%) para 2,5s (fator 70%). Esta situação é diferente daquela do algoritmo com grupos (Figura 6.12), onde o aumento tanto do número de adaptações quanto do fator de quebra provocam aumentos menos significativos no tempo de sobrecarga. Por exemplo, enquanto, para o algoritmo com grupos, o tempo médio de sobrecarga varia entre 350ms e 900ms, de acordo com o número de adaptações (1 a 5), nos fatores de quebra 0.4 a 0.7, o tempo de sobrecarga para o algoritmo ingênuo apresenta uma maior amplitude, variando de 300ms a 2200ms (fator 70%) dependendo do número de adaptações.

Pode-se perceber que o ganho de desempenho do algoritmo com grupos, quando comparado ao algoritmo ingênuo, também é significativo (chegando a 100% em alguns casos).

6.1.7 Pior caso para o algoritmo com grupos

Este último experimento visa avaliar o comportamento do algoritmo com grupos para o seu pior caso, ou seja, quando o número de clientes é inferior à potência de dois do número de situações de contexto ($k < 2^n$), como explicado na Seção 4.3.3, Equação 4-10. Na simulação, a definição de quais clientes necessitam de adaptações, e em que quantidade, foi realizada da mesma forma como descrito na Seção 6.1.4.

As Figuras 6.17 e 6.18 apresentam uma comparação do número de grupos criados, e do tempo de médio de serviço, respectivamente, para os casos onde existem 5, 7 e 10 situações de contexto, e para 500 e 1000 clientes.

A Figura 6.17 demonstra que o comportamento dos grupos formados depende principalmente do número de situações de contexto e do número de clientes, mas também do fator de quebra, que define a porcentagem de clientes ativos por grupo. Pode-se observar, que enquanto para 5 e 7 situações de contexto (casos em que $k > 2^n$) o número de grupos criados atinge o máximo de 32 e 128 (ou seja, 2^n), respectivamente, para 10 situações de contexto, o número de grupos criados é proporcional ao número de clientes (k).

A Figura 6.18 mostra o comportamento do tempo médio de serviço para casos com diferentes quantidades de situações de contexto, e com variação da porcentagem de clientes ativos por grupo. O tempo de serviço apresentado é calculado levando-se em consideração o tempo médio de serviço para cada grupo multiplicado pelo número de clientes atendidos em cada grupo, obtendo-

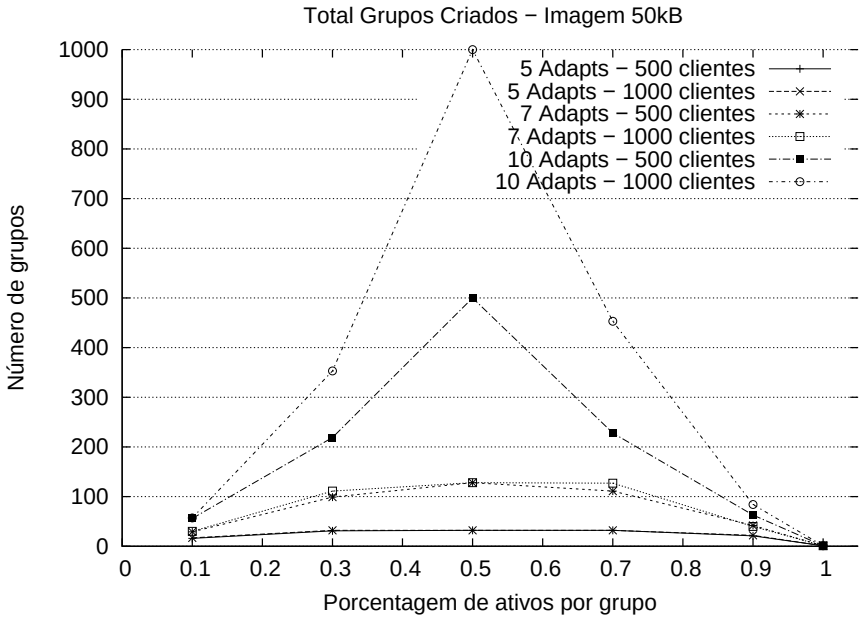

Figura 6.17: Total de grupos criados para 5, 7 e 10 situações de contexto.

se assim uma média ponderada. Como era de se esperar, o aumento do número de adaptações eleva o tempo de serviço. Por exemplo, para 5 situações de contexto, o pior tempo médio de serviço para 1000 clientes chega a 1s, enquanto que para 7 situações, este tempo se eleva para 1,5s.

Nos experimentos realizados, o pior caso do algoritmo com grupos pode ser verificado quando há 10 situações de contexto e uma proporção de 50% de clientes realizando adaptações em cada situação, o que acarreta a geração de um número de grupos igual ao de clientes. O tempo médio de serviço se degrada pois na última situação haverá sempre apenas um cliente por grupo, e a metade destes realizam a mesma adaptação repetidamente. Mas, deve-se notar que para todas as demais situações de contexto anteriores, há compartilhamento de adaptações pelos clientes, e por isso o algoritmo continua sendo muito superior ao algoritmo força bruta.

Vale mencionar ainda que, se o número de clientes aumentar, a curva do tempo médio de serviço ponderado tende a se "suavizar", e apresentar valores consistentemente inferiores aos apresentados na Figura 6.18, demostrando a escalabilidade do sistema. Além disso, este é um caso extremo($n < 2^k$ e 50% dos clientes realizando as n adaptações), que acreditamos, raramente ocorrerá na prática. Além disso, na prática é improvável que haja clientes necessitando de um número tão grande (10 ou mais) de adaptações consecutivas ao mesmo tempo.

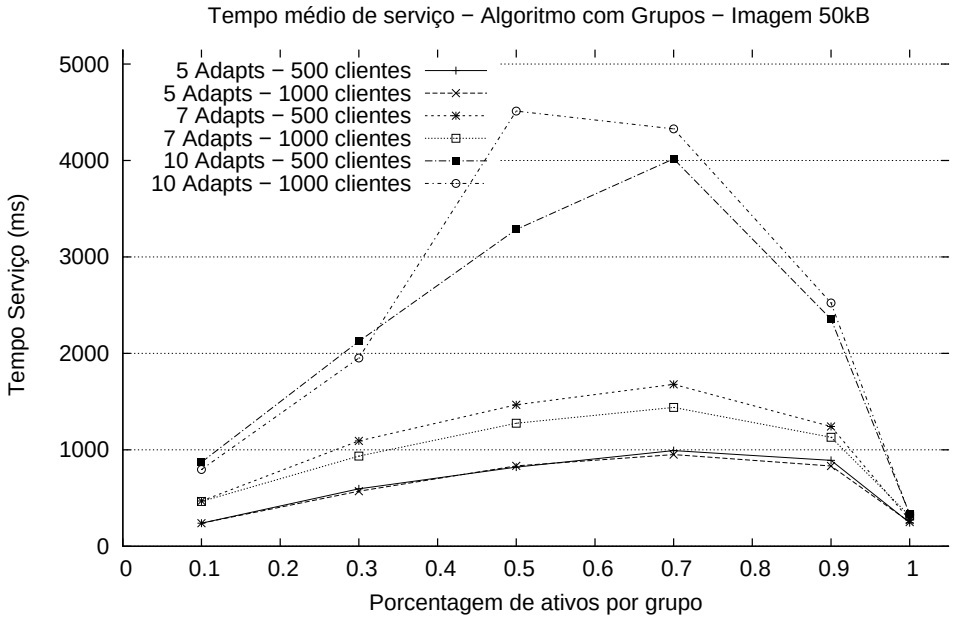

Figura 6.18: Comparação tempo de serviço médio para 5, 7 e 10 situações de contexto

6.2 Avaliação Qualitativa

Sendo o ProxyFramework uma ferramenta de middleware a ser utilizada por desenvolvedores de aplicações, achamos interessante fazer uma avaliação qualitativa sobre a usabilidade da ferramenta. Nesta seção, são apresentados alguns resultados de uma pesquisa sobre a utilização do ProxyFramework para desenvolvimento de aplicações móveis adaptativas, feita com usuários (desenvolvedores). Essa pesquisa foi realizada por meio de um questionário com questões que visaram descobrir se, na percepção dos usuários, a ferramenta proposta apresenta funcionalidades efetivas(i.e, funcionam), usáveis (i.e, fácil compreensão e utilização) e úteis (i.e., atendem às expectativas dos usuários).

Os usuários do ProxyFramework respondentes eram alunos da disciplina Introdução à Computação Móvel, oferecida pelo Departamento de Informática da PUC-Rio no segundo semestre de 2006, que puderam desenvolver aplicações móveis com adaptação de conteúdo como trabalho da disciplina.

Foram desenvolvidas três aplicações similares para um serviço tipo RSS que difunde notícias sobre os fatos jornalísticos mais marcantes do dia. Além de notícias, são enviadas também imagens relacionadas às matérias. Podem se inscrever no serviço clientes utilizando dispositivos como desktops, notebooks, PDAs e celulares. Todas as aplicações eram compostas de três partes: servidor, proxy e cliente, e nas quais o servidor RSS envia mensagens para os clientes

inscritos no serviço periodicamente. As mensagens são então interceptadas por um proxy (instância do ProxyFramework), o qual as repassa para os clientes, efetuando adaptações no conteúdo das mensagens de acordo com o contexto de execução de cada dispositivo.

Os desenvolvedores de cada aplicação criaram um conjunto próprio de adaptadores de texto e de imagem, e definiram diferentes regras de adaptação. Alguns exemplos de adaptadores de texto incluem a extração da manchete (título) da notícia, resumo da notícia, e extração da frase com maior número de palavras. Dentre os adaptadores de imagem desenvolvidos pode-se citar: conversor de colorido para preto-e-branco, redimensionamento pelo tamanho de tela do dispositivo e redução do tamanho da imagem. As regras de adaptação utilizadas incluem contextos estáticos dos dispositivos, como tamanho de tela e tipo de dispositivo (PDA, celular,etc), bem como contextos dinâmicos como quantidade de memória disponível e nível de bateria. As regras de adaptação foram definidas em um arquivo de configuração (como descrito na Seção 5.2). Além disso, cada aplicação implementou uma política de armazenamento de mensagens distinta, para momentos de desconexão, tais como, FIFO e FIFO com timeout. Os *screenshots* dos clientes de uma destas aplicações são mostrados na Figura 6.19.

Após a conclusão do processo de desenvolvimento, foi solicitado aos usuários que respondessem um questionário (descrito no Apêndice D) para se obter indicações qualitativas sobre o benefício percebido pelos usuários com a utilização do ProxyFramework no desenvolvimento de suas aplicações.

Esse questionário continha perguntas sobre o benefício percebido no desenvolvimento das aplicações, sobre aspectos de facilidade de uso, rapidez no desenvolvimento, atendimento às espectativas e pontos a se melhorar. Outras questões serviram para identificar se estavam corretas e eram relevantes algumas das hipóteses sobre os usuários desenvolvedores que nortearam o desenvolvimento do ProxyFramework, tais como:

- O desenvolvedor sabe que tipos de adaptações de conteúdo são razoáveis ou interessantes para cada situação de contexto (dinâmico e/ou estático).
- O desenvolvedor quer ao mesmo tempo flexibilidade para implementar os seus adaptadores e definir o momento em que eles devem ser disparados.
- O desenvolvedor não quer ter que se preocupar com questões de encaminhamento de mensagens individualmente (p.ex. no caso de Pub/Sub)

Essas hipóteses assumem que o usuário deve se preocupar primordialmente com a lógica de negócios de sua aplicação, e é o mais indicado para determinar as situações de contexto de interesse e as adaptações específicas

Figura 6.19: Aplicação exemplo

para cada situação em sua aplicação. De maneira geral, as hipóteses apontadas acima se mostraram bastante condizentes com as necessidades e intenções dos desenvolvedores, e atenderam às espectativas. Os usuários aprovaram a forma como o ProxyFramework foi projetado, como pode ser observado citando partes de suas respostas:

> "... o que eu espero é que ele me ofereça uma plataforma simples para que eu possa concentrar meus esforços na lógica de negócios (adaptadores). E isso o ProxyFramework faz muito bem."

> "A idéia básica do ProxyFramework é bem intuitiva. Considero que o framework foi bem projetado e implementado e trabalha de forma harmônica com os outros componentes da aplicação (cliente e servidor). "

> "Considero o ProxyFramework, e os demais serviços da MoCA, um bom middleware para aplicações móveis e sensíveis ao con-

texto e considero também que ambos (MoCA e ProxyFramework) atingem seus objetivos satisfatoriamente: possibilitar a construção de aplicações sensíveis ao contexto e reduzir o esforço de construção de tais aplicações, respectivamente."

Além disso foi solicitado que os usuários avaliassem alguns aspectos do ProxyFramework, numa escala de 1 a 5 (5= excelente, 4= bom, 3= razoável, 2= deficiente, 1= muito ruim). As pontuações médias dos principais aspectos são apresentadas na Tabela 6.2.

Item	Média
Ganho em produtividade	4,7
Adequação das interfaces e do formato do arquivo de configuração	4,7
Simplicidade/naturalidade dos conceitos	4,3
Facilidade de instalação/configuração	3,5
Facilidade de aprendizado	3,9
Robustez e confiabilidade do ProxyFramework	4,5
Documentação	3,3
Avaliação Geral	4,3

Tabela 6.2: Avaliação dos usuários

As principais dificuldades encontradas pelos usuários no uso do ProxyFramework estavam relacionadas ao ambiente de execução e de testes. Como proxy é um intermediário entre a aplicação e a MoCA, e depende das informações de contexto fornecidas por ela, seu correto funcionamento está atrelado à instanciação de vários elementos distribuídos da arquitetura (tais como: CIS, Monitor). Estas dificuldades não são especificamente do framework, mas sim da complexidade intrínseca da MoCA como um todo, e da interação de seus elementos. Esse problema reflete-se na avaliação do item "Facilidade de instalação/configuração"da Tabela 6.2. Esta dificuldade deve-se em parte à documentação, que se mostrou incompleta e deficiente, e que também impactou na facilidade de aprendizado do framework.

Os itens com melhor pontuação foram aqueles relacionados ao uso do ProxyFramework na implementação da aplicação. As interfaces dos pontos de extensão (adaptadores) e configuração (regras de adaptação) foram consideradas adequadas às necessidades, além de serem simples de compreender e implementar.

O item 'ganho em produtividade' foi o que obteve as maiores notas. Os usuários afirmaram que só puderam desenvolver aplicações adaptativas em um curto espaço de tempo graças ao uso do ProxyFramework. Em média, após o tempo de aprendizado, instalação e configuração da arquitetura MoCA,

o desenvolvimento das aplicações durou menos de três dias. Na opinião dos usuários, o tempo de desenvolvimento teria sido no mínimo 5 vezes maior sem o uso do framework.

Um outro indicativo freqüentemente usado para avaliar o grau de complexidade da implementação das aplicações desenvolvidas é o número de linhas de código (excluindo comentários) implementado pelos desenvolvedores. Esta métrica é mostrada na Tabela 6.3 para cada uma das partes das três aplicações desenvolvidas. Pode-se notar que o desenvolvedor teve um esforço reduzido na implementação da aplicação, podendo se concentrar nas atividades de negócio, como a implementação de adaptadores. Os adaptadores implementados variaram entre 50 e 100 linhas de código. Apenas para comparação, o código do ProxyFramework possui em torno de 5000 linhas.

	App 1	App 2	App 3
Servidor	221	293	185
Cliente	245	311	130
Instância Proxy	91	107	115
Adaptadores	264	204	302
Configuração de Regras (XML)	71	55	74

Tabela 6.3: Métricas das aplicações desenvolvidas com uso do ProxyFramework

De acordo com os resultados desta pesquisa, nota-se uma satisfação do usuário, e pode-se considerar que por isso, o ProxyFramework, como ferramenta de apoio ao desenvolvimento, cumpre seus objetivos, sendo simples de usar, reduzindo o esforço de implementação e propiciando um ganho significativo em produtividade. O ProxyFramework mostrou-se adequado e útil para o desenvolvimento de aplicações adaptativas móveis. Por fim, mais uma citação de um questionário:

> "Honestamente, achei muito interessante o trabalho de adaptação de conteúdo baseado em contexto e uma das coisas que mais me surpreendeu foi a facilidade com que o mesmo foi desenvolvido, graças ao uso do ProxyFramework. "

6.3 Resumo e Conclusões

Neste capítulo, foram apresentados os resultados de testes de desempenho dos algoritmos força bruta, ingênuo e com grupos (*Context-Aware Client Grouping Algorithm*) usados para gerência de adaptadores do sistema proposto, bem como uma avaliação de usuários que utilizaram ao framework para desenvolvimento de aplicações adaptativas móveis.

Nos experimentos foram utilizados cenários de certa forma artificiais pois assumiam condições pouco prováveis em uma situação real, como por exemplo, o compartilhamento de todas as adaptações por todos os clientes. Entretanto tais cenários são necessários para testar o desempenho em situações extremas de execução dos algoritmos.

Os resultados mostraram que o desempenho dos algoritmos com grupos dependem de diversos fatores, em especial, o grau de compartilhamento de adaptações comuns entre os clientes e também a forma como estes clientes são distribuídos na formação dos grupos. De maneira geral, quanto maior o número de clientes com as mesmas características e situações de contexto semelhantes, maior será o número de adaptações compartilhadas, e conseqüentemente, o melhor desempenho do algoritmo com grupos.

Num cenário real, provavelmente, haverá frequentemente a formação de subgrupos de clientes que executam um mesmo subconjunto de adaptadores. Acreditamos que esta situação é bem mais comum do que um cenário em que os clientes demandam a execução dos mesmos adaptadores (mesma seqüência completa) e, assim, o uso do algoritmo com grupos sempre apresentará ganho em relação ao algoritmo ingênuo, e o custo para isso é bastante reduzido.

Não foram efetuados testes para o algoritmo que trata adaptações parametrizadas por contexto. Estas adaptações, utilizam os valores dos atributos de contexto como parâmetros da adaptação e podem ser bastante comuns num cenário real. O desempenho deste algoritmo tende a ser pior que o do algoritmo com grupos original, pois um maior número de grupos é formado, devido às adaptações utilizarem valores absolutos de contexto. Além disso, quanto mais específicas as regras de adaptação, o desempenho aproxima-se do algoritmo força bruta.

Dessa forma, na definição das regras de adaptação, recomenda-se ao desenvolvedor (ou administrador) do sistema adotar uma solução de compromisso entre desempenho e satisfação do usuário. As adaptações devem atender às necessidades dos clientes móveis, mas ao mesmo tempo, é importante que elas sejam compartilhadas por um maior número de clientes, favorecendo a eficiência do sistema. Assim, é mais interessante, do ponto de vista de desempenho, que as regras de adaptação definam níveis (ou faixas) de valores, ao invés de valores absolutos. Por exemplo, uma adaptação que redimensiona imagens para o tamanho de tela do dispositivo, ao invés de adaptar a imagem para cada dispositivo, poderia adaptá-la para o menor tamanho de tela de um conjunto de dispositivos semelhantes, e assim atender satisfatoriamente a todos de uma única vez e com um menor custo de adaptação.

Por fim, pode-se ressaltar que, de acordo com os usuários pesquisados, o

ProxyFramework é considerado uma ferramenta de apoio ao desenvolvimento de aplicações móveis muito útil e simples de usar, além de possibilitar uma redução significativa no esforço e tempo de implementação deste tipo de aplicação.