Adaptação ao Contexto em Sistemas de Comunicação Heterogéneos *

José Mocito Liliana Rosa Nuno Almeida
{jmocito,lrosa,nalmeida}@lasige.di.fc.ul.pt

Hugo Miranda Luís Rodrigues
{hmiranda,ler}@di.fc.ul.pt

Universidade de Lisboa

Resumo

Hoje em dia, as aplicações necessitam de ser concebidas para executar numa diversidade de dispositivos heterogéneos, desde servidores, PCs, computadores portáteis, PDAs, ou mesmo telemóveis. No caso de aplicações distribuídas, cada participante poderá estar em execução num dispositivo diferente. Perante esta diversidade, é cada vez mais importante desenhar e concretizar protocolos de comunicação adaptáveis, capazes de se reconfigurar, não só de acordo com o contexto local, mas também em função do contexto dos restantes participantes.

Neste artigo é apresentada uma moldura de middleware *que facilita o desenvolvimento e execução de protocolos adaptáveis em função do contexto. Esta moldura é ilustrada através de uma aplicação cooperativa, que se executa quer em dispositivos móveis quer fixos, e que beneficia da disponibilidade de uma pilha de comunicação em grupo reconfigurável.*

1 Introdução

Hoje em dia, as aplicações necessitam de ser concebidas para executar numa diversidade de dispositivos heterogéneos, desde servidores, PCs, computadores portáteis, PDAs, ou mesmo telemóveis. No caso de aplicações distribuídas, cada participante poderá estar em execução num dispositivo diferente. Considerem-se por exemplo aplicações cooperativas multi-utilizador tais como os jogos em rede ou as aplicações de suporte à troca de mensagens (“chat”, “messenger”, etc). Nestas aplicações é perfeitamente possível que um ou vários participantes estejam na rede fixa, enquanto outros participantes executam a aplicação através de dispositivos móveis (sublinhe-se que diversos assistentes pessoais e consolas móveis suportam inclusive a interacção de vários utilizadores em modo *ad hoc*).

Perante esta diversidade, é cada vez mais importante desenhar e concretizar protocolos de comunicação adaptáveis, capazes de se reconfigurar, não só de acordo com o contexto local, mas também em função do contexto dos restantes participantes. Ilustramos esta necessidade usando como exemplo uma camada de difusão de mensagens. A necessidade de suportar difusão é comum em aplicações cooperativas multi-utilizador (tais como os jogos on-line). Embora a maioria das concretizações actuais destes jogos se baseiem em arquitecturas cliente-servidor, foram já demonstradas as vantagens de utilizar arquitecturas entre-pares (*peer-to-peer*) para aumentar a sua capacidade de escala e eficiência [15].

*Este trabalho foi parcialmente suportado pelo LaSIGE e pelo projecto FCT INDIQoS POSI/CHS/41473/2001 através de fundos POSI e FEDER.

A camada de difusão de mensagens é um bom exemplo de como a concretização do serviço depende do contexto de execução. Tipicamente, a concretização mais imediata de uma camada de difusão recorre ao envio de uma sequência de mensagens ponto-a-ponto (uma para cada participante do sistema). Esta concretização possui a vantagem de ser bastante genérica, mas pode também ser bastante ineficaz. Se todos os participantes estão sobre a mesma rede local, ou num sistema autónomo com capacidade de suportar difusão-IP (*IP-multicast*), a utilização de endereçamento em difusão permite realizar a disseminação de informação de forma mais eficiente. Se os participantes forem em grande número, em redes de grande escala geográfica, pode ser preferível basear a disseminação em protocolos epidémicos [15]. Se existir uma mistura de participantes na rede fixa e de participantes em dispositivos móveis, os participantes fixos podem assumir um papel de maior relevo na disseminação, poupando recursos aos participantes móveis. Se todos os participantes se executarem sobre dispositivos móveis, o protocolo de difusão poderá utilizar informação acerca da bateria disponível em cada nó para maximizar o tempo de vida da rede [17]. Argumentos semelhantes podem ser utilizados para outras camadas da pilha de protocolos (transporte, filiação em grupo, etc). Como é obvio por este exemplo, a capacidade de adaptar a pilha de protocolos à informação de contexto é uma característica de maior relevo para as aplicações do futuro.

Neste artigo apresentamos uma moldura (do Inglês *framework*) de *middleware* que facilita o desenvolvimento e a execução de protocolos adaptáveis em função do contexto. Esta moldura inclui vários componentes, dos quais destacamos os seguintes: um sistema de captura e disseminação de informação de contexto; uma moldura de suporte à composição e execução de protocolos; e um sistema de controlo e reconfiguração dos protocolos de comunicação. O artigo discute a necessidade e requisitos de cada um destes componentes e apresenta uma primeira concretização de cada um deles. A moldura é ilustrada através de uma aplicação cooperativa, que se executa quer em dispositivos móveis quer fixos, e que beneficia da disponibilidade de uma pilha de protocolos de comunicação em grupo reconfigurável.

O resto do artigo está organizado do seguinte modo. A Secção 2 descreve a arquitectura de suporte à adaptação ao contexto em sistemas de comunicação heterogéneos. Um primeiro protótipo desta arquitectura é descrito na Secção 3. A utilização da arquitectura é ilustrada por uma aplicação de exemplo que é descrita e validada na Secção 4. O trabalho relacionado é discutido na Secção 5. A Secção 6 conclui o artigo e apresenta direcções futuras.

2 Moldura *Morpheus*

Este artigo aborda o problema de construir uma moldura de *middleware* que facilite o desenvolvimento de sistemas de comunicação adaptáveis ao contexto. Antes de descrevermos os componentes desta moldura, discutem-se os requisitos que esta necessita satisfazer.

Como já mencionámos, as características do dispositivo onde a aplicação se executa é um dos aspectos de contexto mais determinantes para a configuração dos protocolos de comunicação. Diferentes tipos de dispositivos possuem diferentes características em termos de bateria, memória disponível, capacidade de processamento, etc. Estas características influenciam o tipo de protocolos a utilizar pelo dispositivo. A literatura é rica em exemplos deste tipo de adaptação [1, 9].

O tipo de adaptação referido anteriormente pode ser satisfeito através da configuração *off-line* de diferentes versões dos protocolos para cada dispositivo (de um modo semelhante à configuração da maioria dos sistemas operativos correntes). No entanto, existem múltiplos aspectos do contexto

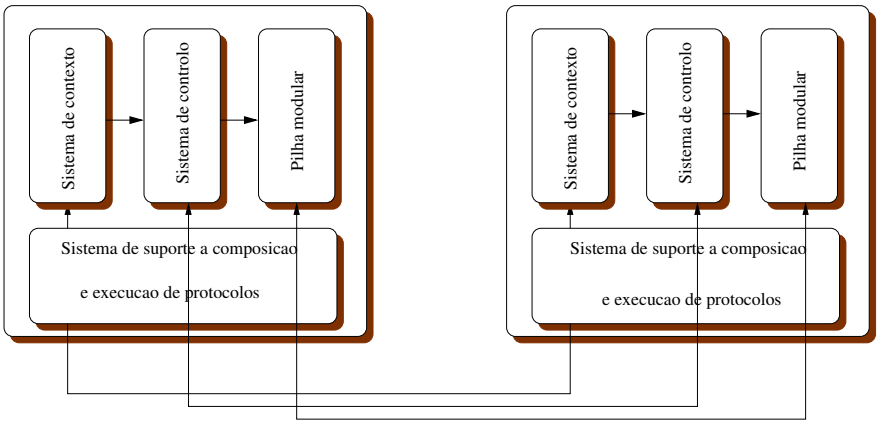

Figura 1: Moldura *Morpheus*

que não podem ser previstos antecipadamente. Por exemplo, a taxa de erros pode influir no tipo de recuperação de erros que é utilizada: para pequenas taxas de erros é preferível detectar os erros e retransmitir as tramas perdidas enquanto para taxas de erro maiores a recuperação para a frente (*Forward Error Correction*) é mais eficaz [11]. Neste caso, a adaptação em tempo de execução é indispensável.

Existem duas aproximações possíveis à construção de sistemas de comunicação adaptáveis, que passamos a descrever de seguida:

- A primeira, certamente mais comum, consiste no desenvolvimento de protocolos adaptáveis monolíticos. Neste tipo de sistema, a lógica de adaptação e reconfiguração está imbricada com os algoritmos de comunicação [16]. A desvantagem desta aproximação é que a totalidade do código necessita de ser carregada em todos os dispositivos. Para além disso, torna-se difícil reutilizar as políticas de adaptação no contexto de outros protocolos.

- A segunda alternativa consiste em utilizar sistemas de comunicação modulares e reconfiguráveis. Segundo esta aproximação, a lógica de adaptação e reconfiguração encontra-se separada dos protocolos de comunicação. Em função da configuração escolhida, executam-se em cada dispositivos apenas os protocolos necessários para suportar a funcionalidade exigida por essa configuração.

O nosso trabalho engloba também outro tipo de adaptação, que consiste na adaptação ao contexto dos participantes remotos. Em particular, para sistemas em que existem participantes na rede fixa e participantes em dispositivos móveis, pretendemos usar protocolos que exploram a maior capacidade (em termos de largura de banda e energia) dos dispositivos fixos. Para suportar este tipo de adaptação é necessário que exista um mecanismo de recolha e disseminação de informação de contexto, que permita à lógica de adaptação obter a informação necessária para desencadear, quando necessário, a reconfiguração do sistema.

Para responder a estes requisitos desenvolvemos uma moldura de *middleware* que chamámos *Morpheus*, ilustrada na Figura 1, que possui os seguintes componentes:

- Um ambiente de suporte à composição e execução de protocolos, que facilita o desenvolvimento de protocolos de comunicação de forma modular.
- Um sistema de recolha e disseminação de informação de contexto.
- Um sistema de controlo e reconfiguração dos protocolos de comunicação.
- Pilhas ou configurações que se adequem aos diversos contextos.

Nos parágrafos seguintes, descrevemos com mais pormenor cada um destes componentes.

2.1 Ambiente de Suporte à Composição e Execução de Protocolos

Os ambientes de suporte à composição e execução de protocolos são molduras de *middleware* que facilitam o desenvolvimento modular de pilhas de protocolos reconfiguráveis. Em particular, estas molduras definem abstracções que facilitam a composição de protocolos, nomeadamente abstracções para suportar a interacção indirecta entre camadas. Quando uma camada pretende comunicar com outra, ao invés de realizar uma chamada directa a uma função da camada adjacente, invoca os serviços dessa camada indirectamente, através de um núcleo de suporte à composição. Esta comunicação indirecta é tipicamente baseada na troca de eventos. Uma vez que as camadas de protocolos comunicam de forma indirecta, torna-se mais simples reconfigurar a pilha de protocolos, removendo ou acrescentando camadas, sem a necessidade de alterar o código de cada um dos componentes. Em tempo de execução, estas molduras oferecem também serviços de suporte tais como gestão de mensagens, gestão de temporizadores, etc.

Podem ser encontrados na literatura vários exemplos de ambientes de suporte à composição de protocolos [7, 2, 6, 12] que poderão ser adaptados para utilização na nossa arquitectura. Uma funcionalidade importante, que facilita a reconfiguração dinâmica, é a possibilidade de transmitir meta-informação sobre a composição de protocolos, entre os diferentes nós do sistema. Em particular, o componente de controlo e reconfiguração deve poder enviar para cada participante informação sobre a composição a usar.

2.2 Sistema de Captura e Disseminação de Informação de Contexto

Um aspecto fundamental na nossa arquitectura consiste na capacidade de capturar informação relevante de contexto, referente a cada participante da aplicação, e disseminar esta informação para a componente de controlo e reconfiguração. Existem diversos desafios que se colocam no desenvolvimento deste sistema.

Em primeiro lugar, o sistema de captura de informação de contexto deve ser facilmente extensível, uma vez que cada pilha protocolar pode necessitar de informação distinta. Como referimos, informação de contexto relevante pode ser o tipo de dispositivo (PC, PDA, telemóvel...), velocidade do processador, dimensão de memória, capacidade do disco, largura de banda, latência na comunicação, entre outras. Alguma desta informação é imutável, outra necessita de ser actualizada periodicamente. Isto significa que o sistema de informação de contexto deve suportar diferentes tipos de sensores.

Em segundo lugar, o sistema de disseminação de informação de contexto deve assegurar que cada componente só recebe a informação de que realmente necessita. Ou seja, a informação de

contexto referente a cada componente não deve ser difundida na totalidade por todos os participantes, pois essa actividade pode consumir de modo excessivo os recursos disponíveis. Pelo contrário, o sistema deve oferecer uma interface do tipo publicação-subscrição (do Inglês, *publish-subscribe)* que permita a cada componente, nomeadamente ao componente de controlo, subscrever apenas a informação relevante para executar as políticas de adaptação definidas.

2.3 Sistema de Controlo e Reconfiguração

Este sistema é responsável por recolher a informação de contexto relevante e por decidir qual a configuração do sistema mais adequada. Deve também ponderar as vantagens que se podem obter por reconfigurar o sistema com o custo da própria reconfiguração de modo a optar pelo momento mais oportuno para proceder à adaptação dinâmica. Por exemplo, perante cenários de conectividade intermitente, a política de adaptação pode ajustar a sensibilidade às alterações de contexto de modo a evitar que o sistema se encontre permanentemente em reconfiguração.

Finalmente, este componente é responsável por coordenar a reconfiguração dos diversos componentes, enviando para cada participante a nova configuração e assegurando que o processo de reconfiguração não introduz incoerências no sistema (para este efeito, o processo de reconfiguração deve assegurar que cada composição está num estado coerente).

Recorrendo ao exemplo do protocolo de difusão de mensagens já referido anteriormente, o sistema de controlo deve actualizar a sua informação sempre que um novo elemento se junta ao grupo ou sempre que um elemento sai do grupo. Em função da composição do grupo (em termos do número de participantes na rede fixa e na rede sem fios) o componente de controlo deve avaliar a necessidade de instalar uma nova combinação de protocolos. Por exemplo, a entrada de um participante fixo pode justificar a reconfiguração da pilha utilizada pelos dispositivos móveis.

Note-se que o sistema de controlo não tem necessariamente de ser centralizado, podendo possuir uma concretização descentralizada, executando-se parcialmente, ou de forma distribuída, em diferentes participantes.

2.4 Comunicação Adaptável ao Contexto

Finalmente são necessários algoritmos/estratégias de comunicação que se adequem da melhor forma aos diversos contextos por forma a tirar partido dos mesmos. Estes algoritmos implementam-se sob a forma de pilhas de comunicação, que prevêm diversos modos de funcionamento.

3 Protótipo

Nesta secção descrevemos o primeiro protótipo da moldura *Morpheus*. O objectivo do desenvolvimento deste protótipo foi facultar uma avaliação preliminar da arquitectura, ilustrando o funcionamento de cada um dos seus componentes e validando a interacção entre os mesmos. Sendo assim, a concretização de cada um dos componentes é necessariamente simplificada: o desenvolvimento de uma versão definitiva de cada um dos componentes é, por si só, um tópico de investigação a prosseguir na continuação deste trabalho. Refira-se que, apesar de simplificada, a concretização de cada componente funciona na sua totalidade e, como será descrito na Secção 4, permite fazer uma validação experimental da arquitectura capaz de demonstrar na prática as suas vantagens.

Na Figura 2 pode observar-se o mesmo diagrama da Figura 1, mas com os diversos componentes

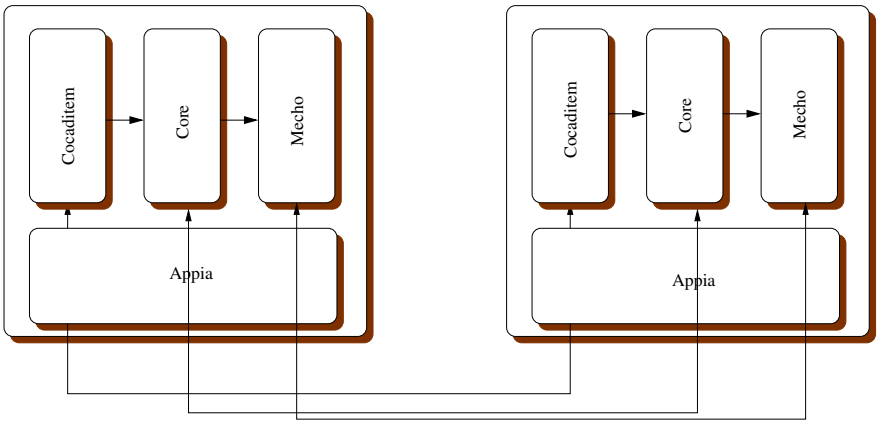

Figura 2: Protótipo

da arquitectura instanciados nas concretizações do protótipo, descritas de seguida.

3.1 Composição e Execução de Protocolos: *Appia*

Começamos por descrever o ambiente de suporte à composição e execução de protocolos usado no nosso protótipo. Existem na literatura diversos sistemas que satisfazem os requisitos descritos anteriormente, incluindo o Coyote/Cactus [2], o Ensemble [6] e o *Appia* [12]. Destes sistemas escolhemos o *Appia* por ser concretizado na linguagem Java, ter sido desenvolvido "na casa" e ser funcionalmente equivalente aos restantes sistemas.

O *Appia* [12] é um sistema modular de suporte à comunicação. Cada módulo do *Appia* é uma camada, i.e. um micro-protocolo responsável por garantir uma determinada propriedade. Estas camadas são independentes e podem ser combinadas. Essa combinação constitui uma pilha de protocolos. Esta pilha de protocolos oferece uma Qualidade de Serviço (QdS) com as propriedades desejadas.

Definida uma QdS é possível criar um ou mais *canais* de comunicação. A cada canal está associado uma pilha de *sessões*: existe uma sessão para cada camada de protocolo cujo objectivo é manter o estado necessário à execução do protocolo da camada correspondente. Dois canais que partilhem um dado protocolo podem partilhar uma sessão. Dado que a sessão é que mantém o estado referente à execução do protocolo, a partilha de uma sessão permite que o protocolo correlacione os eventos trocados nos diversos canais. Por exemplo, se vários canais utilizarem uma mesma sessão de um protocolo causal, todas as mensagens trocadas nesses canais serão ordenadas de forma causal.

A interacção entre camadas é realizada através da troca de eventos. Os eventos são tipificados e cada camada declara ao sistema quais os eventos que cria e que está interessada em processar. O sistema optimiza o fluxo de eventos na pilha de protocolos assegurando que os eventos só são entregues às camadas que registaram interesse no seu processamento.

Uma extensão recente ao *Appia*, realizada pelos autores ao sistema, e de particular relevância para este trabalho, permite construir dinamicamente uma pilha de protocolos com base numa especificação no formato XML [13]. Esta funcionalidade oferece um meio por excelência para

enviar informação sobre as pilhas a utilizar por cada participante quando se executa uma reconfiguração.

A distribuição do sistema *Appia* oferece ainda uma pilha de comunicação em grupo fiável que, como iremos ver, é extensivamente usada no nosso protótipo. Esta pilha é constituída por diversos micro-protocolos que cooperam entre si para oferecer serviços de filiação e comunicação. Em particular o serviço de filiação disponibiliza informação acerca de quais os participantes activos e permite atribuir a cada participante um identificador único, o qual, como veremos, é utilizado na concretização do componente descrito na Secção 3.3.

3.2 Captura e Disseminação de Contexto: *Cocaditem*

O *Co*ntext *Ca*pture and *Di*ssemination Sys*tem* (*Cocaditem*) é, como o nome indica, o componente de captura e disseminação de contexto da nossa moldura. Trata-se de um componente distribuído que é, por sua vez, decomposto num conjunto de sensores, capazes de recolher informação de contexto, e por um sistema de disseminação de informação responsável por distribuir pelos diversos participantes a informação recolhida localmente pelos sensores.

O *Cocaditem* oferece uma interface do tipo publicação-subscrição baseada em tópicos aos seus utilizadores. Os sensores publicam a informação de contexto na forma de 2-tuplos [1] `(itemdesc, valor)` constituídos pela descrição da informação de contexto na forma de sequências de caracteres (por exemplo, "carga no processador", "estado da bateria", "memória disponível", "taxa de erros", etc), e pelo valor correspondente. Por seu lado, os componentes interessados em recolher informação de contexto, subscrevem os tuplos em que estão interessados.

Cocaditem é um componente distribuído. Em cada participante da aplicação executa-se um *agente Cocaditem*. Este agente é responsável por recolher toda a informação capturada pelos sensores locais e por se coordenar com os restantes agentes para assegurar a disponibilidade desta informação junto dos subscritores. A actual concretização do *Cocaditem* é bastante simples. Um canal de controlo baseado num grupo de difusão é utilizado para coordenar todos os agentes. Cada agente limita-se a difundir neste canal toda a informação publicada pelos sensores locais. Isto significa que, no actual protótipo, toda a informação de contexto se encontra duplicada em cada participante. Esta aproximação não possui capacidade de escala e será melhorada em futuras versões do sistema. No entanto, para aplicações que requerem pouca informação de contexto, como é o caso da aplicação de exemplo usada neste artigo, a concretização actual revela um desempenho perfeitamente aceitável.

3.3 Controlo e Reconfiguração: *Core*

Tal como o componente de captura e disseminação de informação de contexto, o componente de *co*ntrolo e *re*configuração (*Core*) também se encontra distribuído. Este componente possui dois sub-componentes: um componente de controlo, responsável por monitorizar o estado do sistema e coordenar a reconfiguração, e agentes locais com a responsabilidade de reconfigurar as pilhas em cada participante. Também à semelhança do *Cocaditem*, os vários agentes do *Core* comunicam através de um grupo de difusão fiável dedicado (de facto, apesar de logicamente independentes,

[1] Esta forma de publicar informação de contexto pretende ser suficientemente simples e clara para ilustrar a nossa arquitectura. Existem, no entanto, melhores formas de publicar esta informação, por exemplo através da utilização de *Type Based Publish/Subscribe* [5] ou até mesmo utilizando XML.

por razões de eficácia, no actual protótipo o *Cocaditem* e o *Core* partilham o mesmo canal *Appia* de comunicação de controlo).

O sub-componente de controlo executa-se com base num coordenador, eleito de forma determinista por todos os membros do grupo de controlo (correntemente, o elemento do grupo de controlo com identificador mais baixo). O sub-componente de controlo obtém informação de contexto através do *Cocaditem* e decide qual a pilha a utilizar por cada participante para transferência de dados, conforme descrito anteriormente.

O processo de reconfiguração executa-se da seguinte maneira. O coordenador envia uma mensagem a todos os agentes locais, de modo a forçar uma mudança de vista na configuração em uso, bloqueando temporariamente o tráfego. Isto assegura que a pilha de comunicação se encontra num estado coerente antes de se aplicar a reconfiguração. De seguida, envia para cada agente a descrição da pilha a instanciar nesse nó. Para isso, é usada a funcionalidade do *Appia* que permite enviar esta informação no formato XML. Cada agente, instancia então a pilha local conforme a descrição XML recebida (isto é trivial usando as funcionalidades oferecidas pelo *Appia*) e a comunicação é reiniciada.

3.4 Algoritmo de Difusão Adaptável: *Mecho*

Para ilustrar a adaptação ao contexto, desenvolvemos uma pilha de protocolos que implementa um algoritmo de difusão reconfigurável que designamos por *M*ulticast *Echo* (*Mecho*). Este protocolo concretiza um serviço de difusão do tipo melhor-esforço, o qual constitui a base de uma pilha de comunicação em grupo fiável oferecendo um leque alargado de serviços incluindo filiação em grupo, difusão fiável e ordenada e sincronia virtual.

Como referimos anteriormente, conjuntamente com o *Appia*, é distribuída uma pilha de comunicação em grupo. A camada de difusão melhor-esforço usada na distribuição base desta pilha concretiza a difusão realizando uma sequência de transmissões ponto-a-ponto, uma para cada participante no sistema. Este é também o algoritmo seguido pelo *Mecho* quando todos os participantes são móveis ou quando todos os participantes se executam em dispositivos fixos. No entanto, em cenários híbridos, com alguns participantes executando-se em dispositivos móveis e outros em dispositivos fixos, o comportamento do *Mecho* altera-se. Nestes cenários, os dispositivos móveis limitam-se simplesmente a enviar uma mensagem para um dos nós fixos que, por sua vez, ecoa estas mensagens para todos os restantes participantes em nós móveis[2].

O protocolo *Mecho* foi construído de maneira modular, possuindo diversas camadas consoante o contexto. Duas configurações possíveis podem ser observadas nas Figuras 3(a) e 3(b).

4 Aplicação de Validação e Resultados Experimentais

Com o objectivo de validar o protótipo descrito na secção anterior foi desenvolvida uma simples aplicação de *chat* que permite comunicar com os elementos filiados num grupo. Uma mensagem enviada por um cliente é recebida por todos os elementos filiados no mesmo grupo que o cliente emissor da mensagem.

[2]É importante ressalvar que esta estratégia funciona quando os nós móveis estão ao alcance de um ponto-de-acesso para a rede fixa. Para outras configurações seria mais eficiente desenvolver outros protocolos adaptativos, inspirados nas ideias descritas em [10].

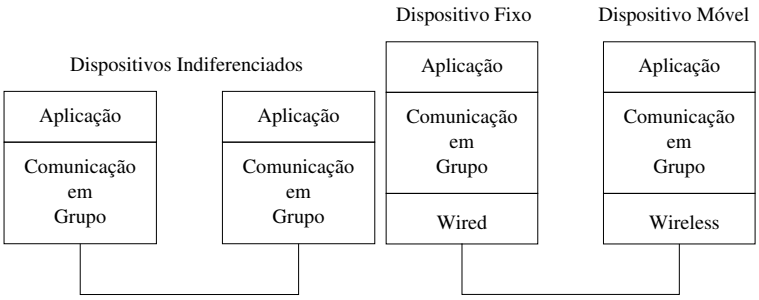

Figura 3: Configurações de pilhas: (a) no caso de dispositivos indiferenciados (b) e adaptadas ao contexto híbrido.

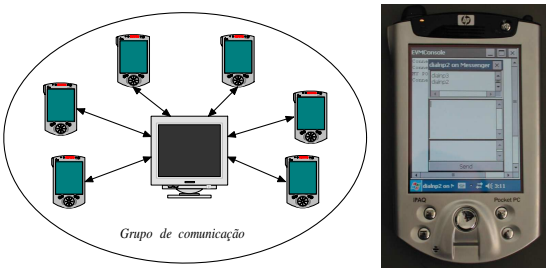

Figura 4: (a) Interacção entre dispositivos; (b) Dispositivo móvel em execução.

Esta aplicação tira partido da pilha de comunicação em grupo fornecida pelo *Appia* para gerir a filiação dos diversos clientes nos grupos. Num ambiente normal, sem propriedades de adaptação, o funcionamento interno da camada de difusão consiste no envio de uma cópia da mensagem para cada elemento pertencente ao grupo. Este comportamento é ideal para validar a ideia por detrás da arquitectura proposta, através da utilização de pilhas reconfiguráveis que tiram partido do contexto em que a aplicação se executa, tal como descrito na Secção 3.4.

A Figura 4(a) apresenta a disposição dos diversos componentes da arquitectura assim como as interacções entre os mesmos necessárias à execução da aplicação (Figura 4(b)) num contexto híbrido, isto é, com a participação de nós móveis e de, pelo menos, um fixo.

4.1 Resultados

Com o objectivo de demonstrar o melhor aproveitamento dos recursos utilizando a moldura *Morpheus*, foi concebido um dispositivo experimental, que tem por base a aplicação de *chat* atrás referida, e cujos resultados devem reflectir esse mesmo aproveitamento, nomeadamente poupando no número de mensagens transmitidas, tendo como consequência a diminuição do gasto energético e de processamento.

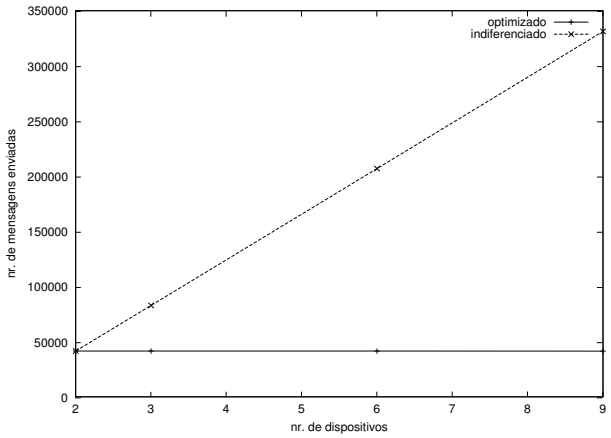

Figura 5: Comparação entre o modo optimizado e indiferenciado

Foram considerados dois cenários distintos: um em que não é retirado partido das características dos diversos dispositivos e um outro onde é aproveitada a existência de um dispositivo fixo. Para cada cenário foi efectuado um conjunto de simulações de dois, três, seis e nove nós. Cada simulação consistiu no envio de dez mensagens de aplicação por segundo até ser atingido o número limite de quarenta mil mensagens. Durante a simulação foram contabilizadas todas as mensagens trocadas pelos dispositivos móveis, que inclui não só as mensagens geradas pela aplicação como também as mensagens de controlo necessárias para assegurar as propriedades de comunicação em grupo.

As simulações nos dispositivos fixos foram efectuadas nos ambientes *Windows* e *Linux*. Como dispositivos móveis foram utilizados Assistentes Digitais Pessoais (PDAs) HP Ipaq 5550, com capacidade de comunicação sem fios 802.11b, a executar o sistema operativo Pocket PC 2003 e o ambiente de execução de aplicações Java *Jeode*.

Os resultados obtidos, expressos na Figura 5, confirmam que a moldura *Morpheus* facilita o desenvolvimento e execução de protocolos adaptáveis em função do contexo. É possível verificar que para o caso em que existem dois nós o número de mensagens contabilizadas é aproximadadamente igual[3], visto que em ambos os casos as mensagens são enviadas para um nó apenas. No entanto, no modo indiferenciado, à medida que o número de nós aumenta, o número de mensagens contabilizadas aumenta linearmente. No modo optimizado, como seria de esperar, o número de mensagens trocadas mantém-se estável, visto que cada nó móvel apenas comunica com um nó fixo. Esta observação indica assim que o ganho em número de mensagens trocadas aumenta com o número de nós intervenientes.

5 Trabalho Relacionado

No âmbito das redes sem fios, muitas têm sido as propostas de sistemas que adaptam os dados às características dos dispositivos ou à qualidade da comunicação. No entanto, raramente é pro-

[3]No modo optimizado o número total de mensagens sofre pequenos aumentos à medida que o número de nós também aumenta, pois são necessárias mais mensagens de controlo relacionadas com a comunicação em grupo.

posta uma moldura de *middleware* que integre a composição de protocolos com a adaptação. Uma arquitectura de referência foi proposta no contexto do projecto MobileMan [4]. Esta arquitectura considera que a informação sobre as condições da rede deve ser disponibilizada a todas as camadas da composição, que podem desta forma cooperar na optimização da utilização dos recursos disponíveis. Para tal, é definida uma área de acesso partilhado, onde todas as camadas publicam e podem obter informações sobre o estado da rede. O MobileMan considera que a adaptação deverá ocorrer individualmente a cada protocolo e não oferece o suporte necessário para a reconfiguração, em tempo de execução, da pilha de composição.

A adaptação dos protocolos de *middleware* tem também sido perseguida através do recurso a linguagens reflexivas como o Iguana/J. O projecto Chisel [8] explora esta aproximação considerando a informação disponibilizada pela aplicação e pelo utilizador para reconfigurar os algoritmos através da reflexividade. Comparativamente, o projecto Chisel adopta uma perspectiva oposta à da nossa proposta por requerer a utilização de uma linguagem especializada e partilha a perspectiva do MobileMan de não suportar a alteração da composição em tempo de execução. Em ambos os casos, a interoperabilidade de dispositivos que requeiram a utilização de protocolos não incluídos no desenho original será bastante dificultada pelas características destes modelos de composição.

O Odyssey [14] é uma plataforma para acesso a dados em dispositivos móveis utilizada para a adaptação ao contexto. O sistema notifica a aplicação quando ocorre uma alteração do interesse dos seus recursos e responde acedendo a diferentes tipos de dados. Este mecanismo desempenha a mesma função que o nosso sistema de controlo e reconfiguração proposto. O projecto Odyssey não suporta concorrência porque não tem uma autoridade central, para resolver problemas relacionados; a nossa proposta, proporciona a utilização de sistemas centralizados ou descentralizados no sistema de controlo e reconfiguração.

O WebPads [3] explora a adaptação em serviços de acesso a Web. Este projecto utiliza troca de eventos de atributos de configuração e utiliza um mecanismo de decisão. Estes processos são semelhantes, respectivamente, aos componentes *Cocaditem* e *Core* do nosso protótipo. O modelo de reconfiguração dinâmica baseia-se num serviço de remoção e adição da configuração dos clientes e servidores. A nossa proposta baseia-se num mecanismo semelhante, utilizando uma extensão do *Appia*, que permite construir dinamicamente uma pilha de protocolos com base numa configuração em formato XML.

6 Conclusões e Trabalho Futuro

O *Morpheus* é uma moldura de *middleware* que facilita o desenvolvimento de sistemas de comunicação adaptáveis ao contexto. Este artigo motiva e descreve esta moldura e cada um dos seus principais componentes. O artigo apresenta também um primeiro protótipo da arquitectura, desenvolvido com o objectivo de validar a interacção entre os componentes e testar o potencial da moldura na optimização do desempenho de aplicações distribuídas em ambiente heterogéneos. Resultados experimentais mostram que, mesmo um protótipo necessariamente simplificado, quando aplicado a redes com um misto de dispositivos móveis e fixos, consegue reduzir o número de mensagens enviadas pelos nós móveis, face a um cenário sem optimizações. Esta redução é tanto mais evidente quanto maior o número de nós participantes. Estes resultados são bastante encorajadores, pelo que no futuro iremos construir versões mais sofisticadas de cada um dos componentes da arquitectura.

Referências

[1] Nalini Moti Belaramani, Cho-Li Wang, and Francis C.M. Lau. Dynamic component composition for functionality adaptation in pervasive environments. In *Proceedings of the Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03)*, pages 226–232, San Juan, Puerto Rico, May 28 – 30 2003. IEEE Computer Society.

[2] N. Bhatti, M. Hiltunen, R. Schlichting, and W. Chiu. Coyote: A system for constructing fine-grain configurable communication services. *ACM Trans. on Computer Systems*, 16(4):321–366, November 1998.

[3] Siu Nam Chuang, Alvin T. S. Chan, Jiannong Cao, and Ronnie Cheung. Actively deployable mobile services for adaptive web access. *IEEE Internet Computing*, 8(2):26–33, March/April 2004.

[4] Marco Conti, Gaia Maselli, Giovanni Turi, and Silvia Giordano. Cross-layering in mobile ad hoc network design. *IEEE Computer*, 37(2):48–51, February 2004.

[5] P. T. Eugster, R. Guerraoui, and J. Sventek. Type-based publish/subscribe. Technical report, Swiss Federal Institute of Technology in Lausanne (EPFL), 2000.

[6] M. Hayden. *The Ensemble System*. PhD thesis, Cornell University, Computer Science Department, 1998.

[7] N. C. Hutchinson and L. L. Peterson. The x-kernel: An architecture for implementing network protocols. *IEEE Transactions on Software Engineering*, 17(1):64–76, 1991.

[8] John Keeney and Vinny Cahill. Chisel: A policy-driven, context-aware, dynamic adaptation framework. In *Proceedings of the IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 3–14, Lake Como, Italy, June4-6 2003. IEEE Computer Society.

[9] Spyros Lalis. Supporting adaptive operation in a dynamically composable personal system. In *Proceedings of the Workshop on European Research on Middleware and Architectures for Complex and Embedded Cooperative Systems (in conjunction with IEEE International Symposium on Autonomous Decentralized Systems)*, Pisa, Italy, April 9–11 2003.

[10] Sung-Ju Lee. *Routing and Multicasting Strategies in Wireless Mobile Ad hoc Networks*. PhD thesis, University of California, Los Angeles, 2000.

[11] M. Luby, L. Vicisano, J. Gemmel, L. Rizzo, M. Handley, and J. Crowcroft. Forward error correction (FEC) building block. RFC 3452, December 2002.

[12] H. Miranda, A. Pinto, and L. Rodrigues. Appia, a flexible protocol kernel supporting multiple coordinated channels. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 707–710, Phoenix, Arizona, April 2001. IEEE.

[13] José Mocito, Liliana Rosa, Nuno Almeida, and Luís Rodrigues. AppiaXML: A brief tutorial. Technical Report DIALNP, Faculdade de Ciências da Universidade de Lisboa, May 2004.

[14] B. D. Noble and M. Satyanarayanan. Experience with adaptive mobile applications in odyssey. *Mobile Networks and Applications*, 4(4):245–254, 1999.

[15] J. Pereira, L. Rodrigues, M. J. Monteiro, R. Oliveira, and A.-M. Kermarrec. Neem: Network-friendly epidemic multicast. In *Proceedings of the 22th IEEE Symposium on Reliable Distributed Systems (SRDS'02)*, pages 15–24, Florence, Italy, October 2003.

[16] L. Rodrigues, S. Handurukande, J. Pereira, R. Guerraoui, and A.-M. Kermarrec. Adaptive gossip-based broadcast. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 47–56, San Francisco (CA), USA, June 2003.

[17] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. Energy-aware wireless networking with directional antennas: The case of session-based broadcasting and multicasting. *IEEE Transactions on Mobile Computing*, 1(3):176–191, jul-sep 2002.