MoCA: Uma Arquitetura para o Desenvolvimento de Aplicações Sensíveis ao Contexto para Dispositivos Móveis *

José Viterbo F., Vagner Sacramento, Ricardo C. A. Rocha, Markus Endler[1]

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

{viterbo,vagner,rcarocha,endler}@inf.puc-rio.br

Abstract. *MoCA is a middleware for developing and deploying context-aware collaborative applications for mobile users. It comprises a service for collecting, storing and distributing context data acquired from mobile devices and also a service for infering the location of such devices. In addition, the architecture provides a set of API's to build applications that exchange context information.*

Resumo. *MoCA é uma arquitetura que oferece recursos para o desenvolvimento e execução de aplicações colaborativas sensíveis ao contexto que envolvem usuários móveis. Esses recursos incluem um serviço para a coleta, armazenamento e distribuição de informações de contexto e um serviço de inferência de localização de dispositivos móveis. Além disso, a arquitetura provê API's para o desenvolvimento de aplicações que interagem com estes serviços.*

1. Introdução

Um desafio para a computação móvel distribuída é dispor de aplicações capazes de perceber e explorar as características dinâmicas do ambiente em que estão inseridas. Para isso é fundamental a existência de uma infraestrutura que permita a essas aplicações tirar proveito dessas características e adaptar seu comportamento de acordo com o contexto percebido [Schilit et al. 1994, Chen and Kotz 2000].

A MoCA (*Mobile Collaboration Architecture*) [Rubinsztejn et al. 2004, Sacramento et al. 2004] é uma arquitetura que oferece suporte ao desenvolvimento de aplicações distribuídas sensíveis ao contexto que envolvem dispositivos móveis interconectados através de redes wireless LAN infra-estruturadas (IEEE 802.11b/g). Os serviços disponibilizados pela MoCA provêem meios para coletar, armazenar e processar informações de contexto obtidas dos dispositivos móveis em uma rede sem fio. Além disso, MoCA integra um conjunto de API's para o desenvolvimento de aplicações que interagem com esses serviços como consumidores de informações de contexto.

Este artigo descreve suscintamente os serviços da arquitetura MoCA e como eles podem ser utilizados no desenvolvimento de aplicações. A seção 2 oferece uma visão geral dos serviços da MoCA e apresenta a arquitetura de uma aplicação típica que interage com esses serviços. A seção 3 fornece maiores informações sobre a

*Trabalho parcialmente financiado pelos projetos CNPq 55.2068/02-2 (ESSMA) e 479824/04-5 (Ed. Universal)

instalação, configuração e utilização dos serviços e API's da MoCA no desenvolvimento de aplicações. A seção 4 apresenta alguns protótipos de aplicações sensíveis a contexto baseadas na arquitetura MoCA. A seção 5 apresenta algumas considerações finais e enumera nossos trabalhos futuros.

2. Visão Geral

Na sua forma mais geral, uma aplicação desenvolvida com base na MoCA é composta por um servidor da aplicação, normalmente executado na rede fixa, e os clientes da aplicação, que são executados em dispositivos móveis. O servidor da aplicação é também um cliente dos serviços MoCA, ou seja, um consumidor de informações de contexto. Ele se registra serviço no `DS` (*Discovery Service*), informando seu endereço e características do serviço, e poderá ser localizado pelos clientes da aplicação. A arquitetura típica de uma aplicação MoCA que adota o modelo cliente/servidor é mostrada na Figura 1.

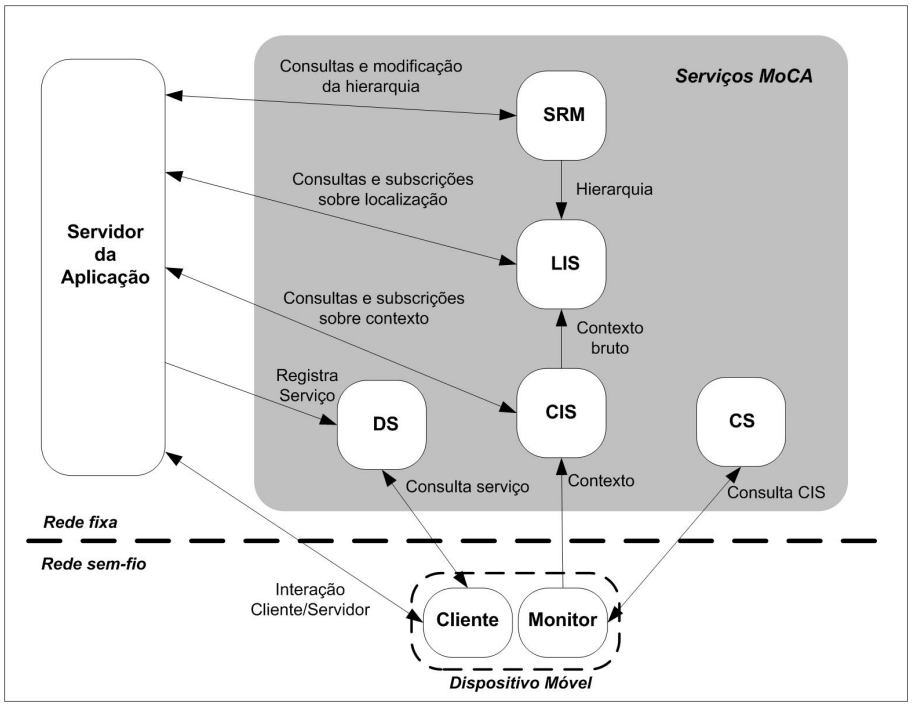

Figura 1. A arquitetura típica de uma aplicação cliente/servidor MoCA

No dispositivo móvel é executado também o `Monitor`, serviço responsável por coletar e divulgar as informações de contexto do dispositivo e da rede. Dentre as informações coletadas estão a qualidade da conexão sem-fio, a carga da bateria, o uso da CPU, a memória livre, o ponto de acesso corrente (AP), e uma lista de todos os pontos de acesso dentro do alcance do dispositivo e a respectiva potência dos sinais recebidos. Tais informações são enviadas periodicamente para o serviço de informação de contexto, o `CIS`, identificado através do serviço `CS` (*Configuration Service*), que disponibiliza informações sobre qual servidor `CIS` é responsável por coletar dados do dispositivo e qual deve ser a periodicidade do envio de dados.

O `CIS` (*Context Information Service*) recebe, armazena e processa as informações de contexto enviadas pelas instâncias do `monitor` em execução nos diversos dispositivos móveis. Estas informações podem ser consultadas pelas aplicações interessadas de

forma síncrona ou assíncrona. Através de consultas síncronas, aplicações podem solicitar informações atualizadas sobre o contexto de um determinado dispositivo. Através de consultas assíncronas, aplicações podem registrar interesse em estados específicos, descritos por expressões lógicas, envolvendo diversas variáveis de contexto de um dado dispositivo.

As informações disponibilizadas pelo `CIS` podem ser utilizadas para derivar informações de contexto de mais alto nível por outro serviço. Por exemplo, o `LIS` (*Location Inference Service*) é um serviço responsável por inferir a localização aproximada de um dispositivo móvel comparando o padrão corrente de sinais de radiofreqüência observados pelo dispositivo (obtidos de pontos de acesso 802.11 dentro do raio de cobertura) com o padrão de sinais medidos em pontos de referência pré-definidos, seja em um ambiente fechado (*indoor*) ou aberto (*outdoor*). O `LIS` permite ao usuário definir regiões simbólicas, ou seja, associar nomes a regiões físicas bem definidas (por exemplo, salas, prédios, corredores), que são de interesse para aplicações sensíveis a localização. As informações do `LIS` podem ser consultadas de forma síncrona ou assíncrona. Na forma síncrona, aplicações podem consultar quais as áreas simbólicas mapeadas no serviço, em que área simbólica se localiza um dado dispositivo e quais dispositivos se encontram em uma determinada área simbólica. Através de comunicação assíncrona, aplicações podem registrar interesse em eventos de mudança de área de um dispositivo específico ou eventos em que qualquer dispositivo entre ou saia de uma dada área simbólica.

Finalmente, o `SRM` (*Symbolic Region Manager*) permite estabelecer uma relação entre as regiões atômicas definidas pelo `LIS`, definindo uma hierarquia em que regiões podem estar subordinadas a outras, ou seja, contidas em outras regiões.

3. Uso da MoCA

Todos serviços e API's da MoCA foram desenvolvidos em Java e podem ser obtidos no endereço `http://www.lac.inf.puc-rio.br/moca`. A única exceção é o `Monitor`, que foi implementado em C++ e executa somente no ambiente operacional do Windows XP. Os serviços e API's da MoCA podem ser divididos em três conjuntos distintos. O primeiro consiste unicamente do `Monitor` e deve ser instalado apenas nos dispositivos móveis. O segundo conjunto compreende os arquivos que implementam os serviços básicos da arquitetura, tais como o `CIS` e o `LIS`. Esses serviços devem ser instalados nas máquinas que compõem o ambiente de execução, normalmente na rede fixa. O terceiro grupo é constituído pelas API's utilizadas na implementação das aplicações, e que precisam ser instaladas somente nas máquinas que integram o ambiente de desenvolvimento.

3.1. Interfaces de Programação

O conjunto de API's oferecidas pela MoCA para desenvolvimento de aplicações compreende três grupos: as API's de comunicação, que fornecem interfaces de comunicação síncrona e assíncrona (baseada em eventos) via UDP e TCP; as API's principais que fornecem interfaces de comunicação com os serviços básicos da arquitetura; e as API's opcionais que facilitam o desenvolvimento de aplicações baseadas na arquitetura cliente/servidor. As características das principais API's são apresentadas a seguir:

- `Communication Protocol` - auxilia o desenvolvimento de aplicações que implementam trocas de mensagens síncronas ou assíncronas usando TCP ou UDP. Esta API é usada pela maioria das implementações dos serviços MoCA;

- `Event-based Communication Interface` - implementa comunicação assíncrona baseada em eventos (*publish*/*subscribe*);
- `CIS Client` - fornece uma interface de comunicação com o serviço `CIS` para permitir a realização de consultas síncronas ou assíncronas sobre informações de contexto dos dispositivos;
- `LIS Client` - fornece uma interface de comunicação com o serviço `LIS` para permitir a realização de consultas síncronas ou assíncronas sobre informações de localização dos dispositivos e áreas mapeadas no serviço;
- `SRM` - fornece uma interface de comunicação com o serviço `SRM` para definir ou consultar a hierarquia estabelecida entre as regiões atômicas.

Utilizando a API `CIS Client`, uma aplicação pode obter informações correntes de contexto referentes a um determinado dispositivo. O Algoritmo 1 exemplifica uma consulta síncrona ao serviço `CIS` no endereço IP "139.82.24.239" para obter o contexto do dispositivo de endereço MAC "00:02:2D:A5:06:46".

Algoritmo 1 : Exemplo de interação síncrona com o CIS

```
1  InetSocketAddress server = new InetSocketAddress(''139.82.24.239'', ''55001'');
2  Request request = new Request(''00:02:2D:A5:06:46'');
3  tcpClient = new TCPConnection();
4  tcpClient.open(server);
5  tcpClient.send(request);
6  reply = (Reply) tcpClient.nonBlockingReceive(3000);
7  tcpClient.close();
8  ctx = (ContextInformation) reply;
9  deviceCTX = ctx.getDvcContext();
10 DeviceCtxManagement.printOutDeviceContext(deviceCTX);
```

Através de comunicação assíncrona esta API permite à aplicação registrar interesse em estados específicos — descritos por expressões lógicas envolvendo diversas variáveis de contexto de um dado dispositivo — para ser notificada quando o estado é observado. Uma expressão de interesse poderia ser, por exemplo, {`"FreeMem` $<$ `10KB" OR "APChange` $=$ `True"`}, que descreve o estado em que o dispositivo tem memória livre menor que 10KB ou ocorreu uma mudança no ponto de acesso à rede.

O Algoritmo 2 mostra uma consulta assíncrona em que a aplicação solicita ao serviço `CIS` no endereço IP "139.82.24.239" ser notificada quando a carga da bateria do dispositivo de endereço MAC "00:02:2D:A5:06:46" atingir valor inferior a 30% de sua capacidade.

Algoritmo 2 : Exemplo de interação assíncrona com o CIS

```
1  InetSocketAddress server = new InetSocketAddress(''139.82.24.239'', ''55000'');
2  InetSocketAddress local = new InetSocketAddress(''localhost" , ''5000'');
3  CisSubscriber subscriber = new CisSubscriber(ECIClient.TCP, server, local);
4  Topic topic = subscriber.subscribe(''00:02:2D:A5:06:46'', ''EnergyLevel < 30'');
5  CISListener listener = new CISListener(''carga baixa'');
6  subscriber.addListener(listener, topic);
```

Na utilização da API `LIS Client` uma aplicação pode realizar consultas síncronas ao `LIS` para obter informações específicas sobre áreas ou dispositivos, ou consultas assíncronas, para registrar interesse em eventos de mudança de área de um dispositivo específico ou eventos em que qualquer dispositivo entre ou saia de uma dada área simbólica.

Algoritmo 3 : Exemplo de interação com o LIS

```
1   LocationInferenceService lis = null;
2   lis = new LocationInferenceService(''locahost'', ''55021'', ''55020'', ''5000'', ''TCP'');
3   allregions = lis.getAtomicRegions();
4   String [ ] areas = new String [allregions.length];
5   for (int i = 0; i < allregions.length; i++) areas [i] = allregions [i].getName();
6   alldevices = lis.getDevices();
7   region = lis.getRegion(''00:02:2D:A5:06:46'');
8   devices = lis.getDevices(''Sala 201'');
9   DeviceListen deviceListen = new DeviceListen();
10  lis.subscribe(''00:02:2D:A5:06:47'', deviceListen);
11  RegionListen regionListen = new RegionListen();
12  lis.subscribe(''Sala 202'', regionListen);
```

O Algoritmo 3 mostra uma consulta síncrona de uma aplicação ao `LIS` (em execução no endereço local) sobre as áreas simbólicas mapeadas no serviço (Linha 3), os dispositivos sendo acompanhados pelo serviço (Linha 6), a área simbólica em que se localiza o dispositivo de endereço MAC “00:02:2D:A5:06:46” (Linha 7), e quais dispositivos se encontram na área simbólica de nome “Sala 201” (Linha 8). Através de comunicação assíncrona, a aplicação registra interesse em eventos de mudança de área do dispositivo de endereço MAC “00:02:2D:A5:06:46” (Linhas 9 e 10), eventos em que qualquer dispositivo entre ou saia da “Sala 202” (Linhas 11 e 12).

3.2. Configuração dos Serviços Básicos

O usuário deve configurar e executar os serviços básicos do MoCA antes de testar e colocar em funcionamento sua aplicação. Entretanto, há uma série de dependências funcionais entre os serviços que o usuário deve observar antes de iniciar cada um deles. A Figura 2 mostra um grafo onde os vértices representam cada operação que o usuário deve realizar para colocar sua aplicação em execução. estas operações consistem em escrever arquivos de configuração adequados ou iniciar os serviços.

Cada aresta do grafo indica uma relação de dependência entre os vértices sucessores e predecessores. Por exemplo, antes de iniciar o `CIS`, é necessário escrever seu arquivo de configuração. Antes de iniciar o `LIS` é necessário escrever seus quatro arquivos de configuração, e iniciar o `CIS` e o `SRM`. Para que o usuário possa executar sua aplicação corretamente, todas as operações descritas no grafo devem ser realizados em uma seqüência que satisfaça a uma ordenação topológica do grafo.

3.3. Como Simular um Dispositivo Móvel

Se o usuário pretende testar sua aplicação usando apenas computadores fixos, mas deseja simular um ou mais dispositivos móveis ele pode utilizar o `Monitor/Sim` (*Monitor Simulator*). Essa aplicação simula o comportamento do `Monitor` em execução em um dispositivo móvel, publicando no `CIS` os dados de contexto obtidos de um arquivo de configuração.

O arquivo de configuração do `Monitor/Sim` contém informações tais como o endereço do servidor `CIS`, o intervalo de envio de dados e nomes dos arquivos que contêm os dados que simulam as informações de contexto coletadas pelo `Monitor`. Cada arquivo de *scans*, por sua vez, contém dados sobre uso da CPU, nível de memória, MAC address, sinais de RF, etc, simulando aqueles coletadas em um dispositivo móvel.

Para não ter que preencher manualmente os arquivos de *scans*, o usuário pode configurar o `Monitor/Sim` utilizando uma interface gráfica que oferece uma forma mais

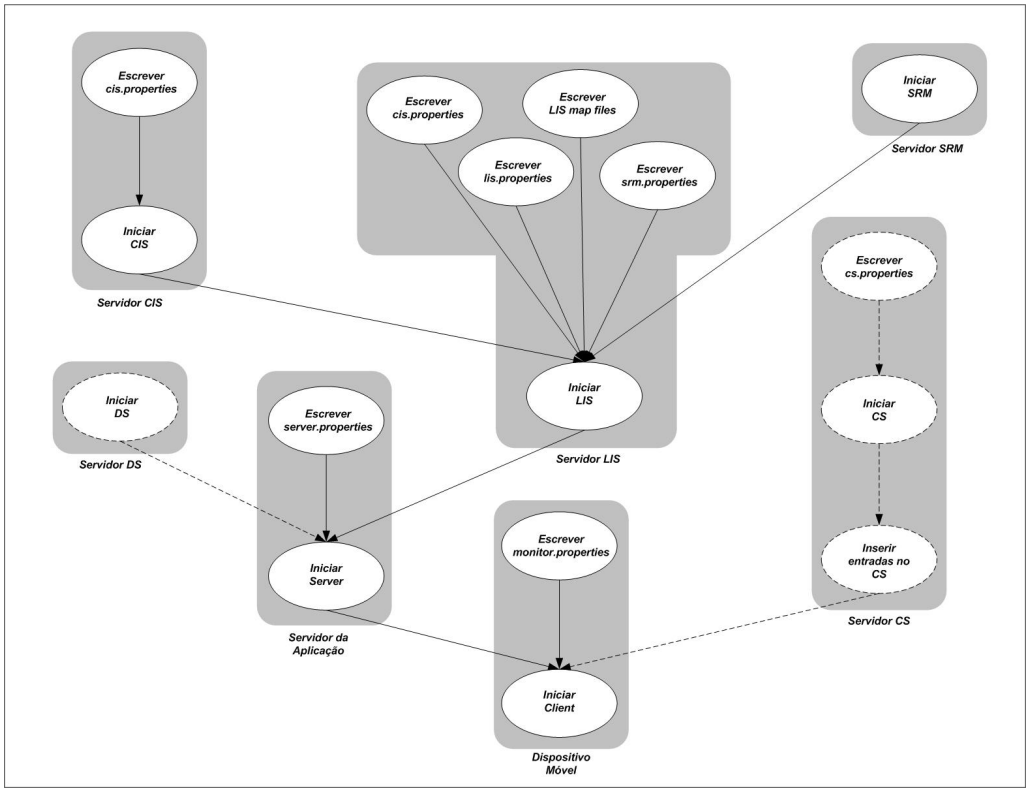

Figura 2. Operações a serem concluídas antes de executar uma aplicação MoCA.

prática de descrever o comportamento do dispositivo móvel sendo simulado. A Figura 3a mostra a janela exibida inicialmente. Para adicionar um arquivo de *scans*, basta pressionar o botão "Adicionar" e selecionar o arquivo desejado. Depois de selecionar o arquivo, ele aparecerá na caixa de seleção "Arquivos" (Figura 3b). Para iniciar o simulador, pressione o botão "Iniciar". Para configurar um arquivo de *scans*, selecione o arquivo na caixa e pressione o botão "Configurar". A janela `Scan Configurator` (Figura 3c) permite ao usuário escrever facilmente todas as informações necessárias para preencher um arquivo de *scans* usado pelo `Monitor/Sim`. Entretanto, é bastante recomendável que o usuário edite os arquivos de *scasn* fornecidos com o pacote disponível para *download*, em vez de escrever arquivos de *scans* totalmente novos para testar suas aplicações.

4. Aplicações Protótipos

Diversas aplicações sensíveis ao contexto foram desenvolvidas utilizando a arquitetura MoCA. Em particular, destacam-se algumas aplicações que utilizam os recursos do `LIS` para implementar serviços baseados em localização, como o `uGuide`, o `Nita` e o `WMS`.

O `uGuide` (*Ubiquitous Guide*) é uma aplicação cliente/servidor que associa uma URL a cada região simbólica registrada no servidor da aplicação. Desta forma, cada vez que um cliente entra em uma certa região, uma pequena tela de *popup* surge acima da barra de tarefas para permitir que o usuário abra em seu navegador de preferência a página correspondente a URL associada àquela região.

O `Nita` (*Notes in the Air*) [Gonçalves et al. 2004] permite a publicação de mensagens (e arquivos em geral) em regiões simbólicas, como se fossem quadros virtuais.

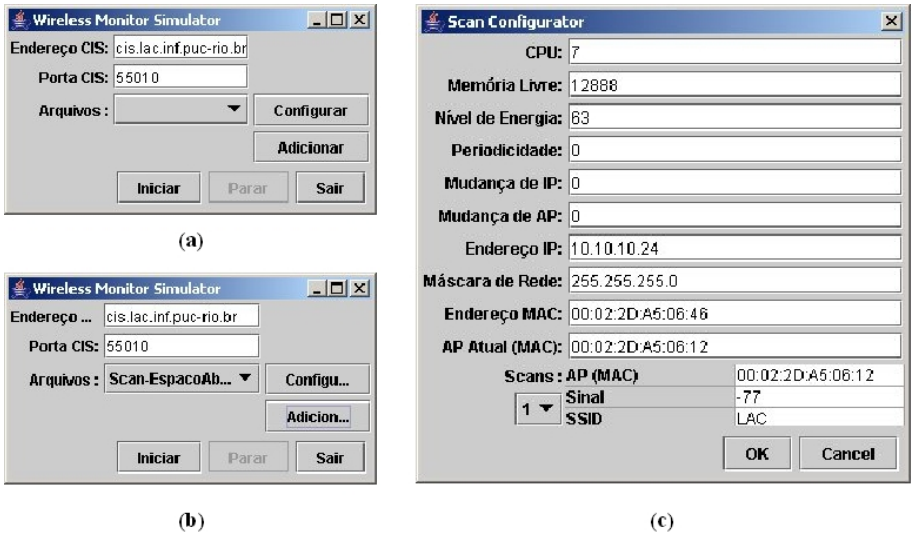

Figura 3. Janela inicial (a), seleção de arquivo (b) e janela de configuração (c)

Desta forma, qualquer cliente que está (ou entra) em uma região simbólica onde "habita" uma mensagem, e que tem a autorização adequada, será notificado sobre a mensagem, e será capaz de lê-la e salvá-la em seu dispositivo. Essa aplicação também permite comunicação síncrona baseada em localização, isto é, o estabelecimento de salas de bate-papo definidas para regiões simbólicas. Desta maneira, somente usuários dentro de uma determinada região simbólica (por exemplo, uma sala de aula) poderão participar da conversa. E toda vez que um usuário deixar a região fisicamente, ele será removido da sala correspondente.

O `WMS` (*Wireless Marketing Service*) é uma aplicação que possibilita que estabelecimentos comerciais enviem anúncios e cupons de desconto direcionados a um local específico para clientes móveis que passeiam em um shopping ou uma loja de departamentos. A aplicação provê interfaces *online* tanto para o registro de uma nova campanha de marketing pela empresa (definindo, por exemplo, a duração, clientes alvo, etc), quanto para o registro de perfis de compra ou *anti-spam* pelos usuários. Para cada campanha de marketing, a companhia pode escolher uma ou mais regiões simbólicas do `LIS` nas quais usuários (isto é, dispositivos móveis) devem ser detectados. Toda vez que um cliente em potencial (predisposto a receber cupons) entra em uma destas regiões, o servidor da aplicação cria e envia para o cliente em execução no dispositivo móvel um cupom de marketing virtual que permanece válido durante todo o período pre-estabelecido. Esse cupom é específico para um determinado usuário e não pode ser replicado.

5. Conclusão e Trabalhos Futuros

A MoCA oferece suporte ao desenvolvimento de aplicações distribuídas sensíveis ao contexto que envolvem dispositivos móveis interconectados através de redes 802.11. Os serviços providos pela MoCA livram o programador da obrigação de implementar serviços específicos para a coleta e tratamento de contexto. Além disso, as API's disponíveis para o desenvolvimento de aplicações simplificam bastante seu trabalho.

A fim de adicionar à MoCA outras funcionalidades importantes, novos serviços e módulos estão em desenvolvimento. Um serviço de privacidade, chamado `CoPS` (*Context*

Privacy Service), foi projetado e está sendo implementado para controlar como, quando e a quem devem ser fornecidas informações de contexto [Sacramento et al. 2005]. O `CoPS` é um serviço opcional que permite ao usuário de aplicações sensíveis ao contexto ou baseadas em localização definir e gerenciar suas políticas de privacidade em relação às suas informações de contexto. Antes de atender a qualquer solicitação de informações de contexto, os serviços MoCA consultariam o `CoPS` para decidir se o acesso àquelas informações deve ser concedido ou negado. As principais funcionalidades oferecidas pelo `CoPS` são controle de acesso baseado em grupos, políticas de acesso pessimista, otimista e interativa, para o controle de acesso, regras de privacidade hierárquica e análise da especificidade de regras.

O `ProxyFramework` [Rubinsztejn et al. 2005] visa oferecer recursos de adaptação de conteúdo a aplicações envolvendo dispositivos móveis. Aplicações clientes em execução nesses dispositivos podem estar sujeitas a condições adversas, como, por exemplo, baixa qualidade de conexão à rede ou poucos recursos computacionais disponíveis. O *framework* permite criar instâncias de *proxies* que são capazes de interceptar as mensagens trocadas entre servidores e clientes móveis e executar transformações em seu conteúdo para adequá-lo às condições de contexto correntes.

Referências

Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research. Technical Report TR2000-381, Department of Computer Science, Dartmouth College.

Gonçalves, K., Rubinsztejn, H. K., Endler, M., Silva, B. S., and Barbosa, S. (2004). Um aplicativo para comunicação baseada em localização. In *Anais do Workshop de Comunicação sem Fio e Computação Móvel*.

Rubinsztejn, H., Endler, M., and Rodrigues, N. (2005). A framework for building customized adaptation proxies. In *Proc. of the IFIP conference on Intelligence in Communication Systems (INTELLCOMM 2005)*.

Rubinsztejn, H. K., Endler, M., Sacramento, V., Gonçalves, K., and Nascimento, F. N. (2004). Support for context-aware collaboration. *First International Workshop on Mobility Aware Technologies and Applications (MATA 2004)*, 5(10):34–47.

Sacramento, V., Endler, M., and do Nascimento, F. N. (2005). A privacy service for context-aware mobile computing. In *Proc. of the IEEE Conference on Security and Privacy for Emerging Areas in Communication Networks*, pages 182–193.

Sacramento, V., Endler, M., Rubinsztejn, H. K., Lima, L. S., Gonçalves, K., Nascimento, F. N., and Bueno, G. A. (2004). MoCA: A middleware for developing collaborative applications for mobile users. *IEEE Distributed Systems Online*, 5(10).

Schilit, B. N., Adams, N. I., and Want, R. (1994). Context-aware computing applications. *5th Workshop on Mobile and Ubiquitous Information Access*.