2
Infra-estrutura para provisão de contexto

O uso de dispositivos portáteis e a facilidade de mobilidade dos usuários nas redes sem fio têm aumentado o interesse por aplicações sensíveis ao contexto. Os desenvolvedores destas aplicações geralmente exploram aspectos/propriedades do ambiente operacional (i.e., contexto computacional, pessoal e físico) para oferecer serviços customizados ao usuário, mudar o comportamento da aplicação de acordo com contexto inferido (e.g., localização corrente do usuário) ou disparar alguma adaptação para lidar com as limitações ou variações do ambiente computacional.

Com intuito de oferecer uma infra-estrutura que auxiliasse o desenvolvimento de tais aplicações, trabalhamos na definição e no projeto de uma arquitetura de provisão de contexto chamada MoCA. Além disso, implementamos alguns serviços que constituem o núcleo desta arquitetura. Estes realizam a coleta, o processamento e a divulgação do contexto computacional dos dispositivos móveis e da rede sem fio IEEE 802.11. Os serviços de contexto da MoCA também serviram de base para o desenvolvimento da nossa pesquisa sobre privacidade, pois a partir deles pudemos ter um ambiente de experimentação real para desenvolver e usar aplicações sensíveis ao contexto, identificar e analisar certos riscos de privacidade associados às informações de contexto fornecidas pela arquitetura, e desenvolver e testar o serviço de privacidade proposto.

Neste capítulo, apresentamos uma discussão geral sobre sistemas sensíveis ao contexto e à localização, descrevemos uma visão geral da arquitetura MoCA e das características de seus serviços de provisão de contexto computacional (CIS) e de localização (LIS) e, por fim, apresentamos as discussões finais.

2.1
Sistemas sensíveis ao contexto

Percepção de contexto (*Context awareness*) tem sido apontada como um dos principais paradigmas de programação de aplicações distribuídas para redes móveis. Existem várias definições na literatura sobre "contexto" e "percepção de contexto". Por exemplo, Dey (57) define que: "Contexto é qualquer informação que possa ser utilizada para caracterizar uma situação

de uma entidade considerada relevante para a interação entre um usuário e uma aplicação, incluindo o usuário e a própria aplicação". Schilit (58, 59), um dos precursores da pesquisa da computação sensível ao contexto, categorizou a informação de contexto em três classes gerais:

Contexto computacional: rede, conectividade, custo da comunicação, banda passante, retardo de transmissão, etc.;

Contexto do usuário: perfil do usuário, localização, velocidade de movimentação, proximidade entre as pessoas, situação social, etc.;

Contexto físico: luminosidade, nível de ruído, temperatura, umidade, etc.

Além disso, outros pesquisadores também incluíram nessa relação o contexto temporal caracterizado pela hora do dia, informações de calendário, semana, estação do ano, dentre outros.

Dentre os diversos tipos de contexto existentes, localização é um dos que mais tem atraído a atenção de diversos grupos de pesquisa. Considerando que para algumas aplicações LBS é essencial ter inferência confiável e precisa sobre localização, um sistema de rastreamento confiável é necessário, e é o ponto crítico a ser tratado no desenvolvimento dessas aplicações (1). A seguir, são discutidas algumas abordagens relacionadas ao desenvolvimento de sistemas de rastreamento de localização em ambientes abertos (*outdoors*) e em ambientes fechados (*indoors*).

2.1.1 Sistemas de posicionamento

O sistema de inferência de localização *outdoor* mais comumente utilizado é o *Global Positioning System* (GPS) (60, 61). A partir desse sistema é possível inferir as coordenadas geográficas (altura em relação ao nível do mar, latitude e longitude) que representam a localização do dispositivo (equipado com um receptor de sinal GPS) com uma precisão aproximada de 3 metros. Além do GPS, podem ser utilizadas outras abordagens tais como triangulação dos pontos de acesso em redes GSM (*Global System for Mobile Communications*) ou IEEE 802.11. Entretanto, devido às possibilidades de interferência e variação de sinal nessas redes, a inferência não é tão precisa e confiável como nos sistemas GPS.

No entanto, a inferência da localização baseada no GPS não funciona apropriadamente para ambientes *indoors*, pois a força/intensidade do sinal emitido pelos satélites não é suficiente para penetrar na maioria dos prédios.

Vários projetos de pesquisa desenvolveram seus próprios sistemas de rastreamento/localização para redes *indoors*. O sistema *Active Badge Location* (62) utiliza uma infra-estrutura de sensores de infravermelho (IR) para inferir a localização dos usuários, enquanto o *Active Bat* (63) e o *Cricket Location System* (64) utilizam pulsos ultra-sônicos[1] e sinais de rádio freqüência. A inferência é realizada a partir da localização dos sensores que emitem sinais de IR ou pulsos ultrasônicos. Esses são colocados estrategicamente em determinadas localidades. Nesses sistemas, ao invés do sistema inferir a localização do usuário, o próprio dispositivo determina sua localização a partir da identificação e da localização do sensor no mapa de regiões das áreas mapeadas pelo sistema de posicionamento. Outros projetos, tais como *MiddleWhere* (65), *Ekahau* (66, 67), *Place Lab* (68, 69), *Herecast* (70), *Radar* (71, 72) inferem a localização corrente do dispositivo a partir de técnicas de triangulação dos sinais dos pontos de acesso no raio de cobertura do dispositivo móvel em redes IEEE 802.11.

Conforme discutido em (1), não existe uma maneira uniforme de rastrear localização dos dispositivos com granularidade fina que funcione em ambientes *indoors* e *outdoors*. Na prática, um sistema pode consultar diferentes serviços de localização para localizar diferentes tipos de objetos[2], enquanto um sistema de posicionamento específico pode localizar um objeto através de diferentes técnicas. Por exemplo, usar GPS em redes *outdoors* e, usar redes 802.11, câmeras de reconhecimento facial, cartão de identificação pessoal, dentre outros em redes *indoors*. Entretanto, em todos os sistemas há uma margem de erro com relação à confiabilidade da inferência devido ao ruído presente no sinal, erros dos sensores, alta taxa de mobilidade dos usuários, etc.

2.2 Desenvolvimento de aplicações sensíveis ao contexto

O desenvolvimento de aplicações sensíveis ao contexto ainda apresenta vários desafios por causa da complexidade em coletar, processar, modelar as informações de contexto e interpretá-las de forma apropriada para adaptar o comportamento da aplicação. A abordagem na qual o desenvolvedor também precisa se preocupar com a captura e processamento da informação contexto não favorece a reutilização de código e não oferece abstrações de programação para o desenvolvimento de aplicações sensíveis ao contexto. Além disso, essa abordagem demandaria mais tempo, mais experiência e esforço de programação para o desenvolvimento da aplicação.

[1] Os ultrasons são ondas acústicas com freqüências acima do limite audível. Normalmente, as freqüências ultrasônicas situam-se na faixa de 0,5 a 25Mhz.

[2] objeto pode ser um usuário, um dispositivo ou um serviço provido pela rede

Uma tendência comum para resolver essas questões visa a separação de tarefas, onde a inferência e o processamento da informação de contexto são realizadas de forma transparente para o desenvolvedor da aplicação, sendo essas tarefas tratadas dentro de uma camada de *middleware* (24). Por exemplo, o desenvolvimento de aplicações LBS em redes IEEE 802.11 se torna uma tarefa muito complexa devido à dificuldade de coletar os sinais de RSSI (*Received Signal Strength Indicator*) dos *Access Points* (APs), processar tais informações e inferir a localização do usuário[3]. No entanto, tal complexidade pode ser atenuada a partir de uma infra-estrutura que contemple os serviços que desempenham as tarefas necessárias para a inferência da localização. De uma forma geral, o *middleware* seria responsável por coletar e agregar as informações de sensores e distribuí-las para as aplicações interessadas. Com esse intuito, projetamos a arquitetura MoCA para oferecer ao desenvolvedor uma infra-estrutura de *software* que trate da complexidade envolvida na coleta e difusão da informação de contexto computacional e de localização.

2.3 Infra-estrutura de contexto

A fase inicial desta pesquisa consistiu do projeto e especificação da arquitetura MoCA (*Mobile Collaboration Architecture*), e implementação de alguns de seus serviços e APIs. Estes são utilizados para a coleta e distribuição de informações de contexto da rede e dos dispositivos, e, em particular da intensidade do sinal de rádio freqüência (RF) dos pontos de acesso que formam a base para a inferência de localização baseada em RSSI. Dentre os serviços e APIs implementadas, estão as APIs *Communication Protocol* (para comunicação síncrona e assíncrona via UDP e TCP) e *Event-based Communication Interface* (ECI) (73) (para comunicação baseada em eventos) e, em especial, o agente para coleta de contexto do dispositivo e da rede (Monitor) e o serviço de informação de contexto (CIS). Além desses, a arquitetura MoCA oferece outros serviços desenvolvidos como resultado da pesquisa de mestrado e doutorado de alguns membros do laboratório LAC (*Laboratory for Advanced Collaboration*). Por exemplo, um serviço de inferência de localização (*Location Inference Service* - LIS) (5), um framework com suporte à adaptação de conteúdo (Proxy-Framework) (74) e um serviço de descoberta (*Discovery Service - DS*). Uma descrição mais detalhada sobre a arquitetura MoCA está disponível em (3, 2).

No projeto dessa infra-estrutura de provisão de contexto, trabalhamos na definição dos papéis, das interfaces e protocolos de interação/comunicação,

[3]Entenda-se por localização do usuário, a localização do dispositivo que o usuário está utilizando.

e na implementação dos serviços que fazem parte do núcleo da MoCA.

As APIs de comunicação, o Monitor e CIS não implementam diretamente a inferência da localização, no entanto, serviram como base para o desenvolvimento do LIS e dos demais serviços da arquitetura. Além disto, o CIS e o Monitor também podem ser acessados diretamente por aplicações e serviços sensíveis ao contexto que necessitem obter o contexto corrente do dispositivo (por exemplo, o nível da sua bateria) e da rede sem fio.

Para tratar das necessidades de privacidade dos usuários das aplicações sensíveis ao contexto, estamos propondo como principal contribuição deste trabalho um serviço de privacidade (CoPS) que determinará como as informações de contexto processadas e manipuladas pelo LIS e CIS serão divulgadas.

A nossa proposta está focada em tratar das questões de privacidade das *informações de localização* do usuário (fornecidas pelo LIS) porque não está claro de que forma tais questões se aplicam também ao acesso às informações de contexto computacional do dispositivo e da rede, tais como: uso da CPU, nível de energia, memória disponível, endereço IP. No entanto, do ponto de vista arquitetural e das interfaces, o CoPS pode também ser integrado a um serviço de provisão de contexto computacional tal como o CIS. De fato, tal integração já foi implementada para averiguarmos as dificuldades envolvidas nesse processo. Porém, não fizemos um estudo para analisar a real necessidade do controle de privacidade das informações providas pelo CIS.

O estado atual dos serviços de provisão de contexto da MoCA é descrito brevemente nas seções seguintes.

2.4 MoCA - Mobile Collaboration Architecture

A arquitetura MoCA (2, 3) auxilia o desenvolvimento de aplicações colaborativas móveis através de serviços de coleta, agregação e difusão de informações de contexto computacional e de localização. A MoCA consiste de serviços de provisão contexto que disponibilizam para as aplicações o contexto da rede e do dispositivo, de APIs de comunicação e um framework (ProxyFramework) para auxiliar a integração das aplicações com tais serviços através de uma comunicação síncrona ou assíncrona (baseada em eventos). Uma visão geral dos serviços da MoCA é ilustrada na Figura 2.1.

No projeto da MoCA, definimos uma separação de tarefas em serviços distintos, onde cada serviço implementa uma funcionalidade específica para a coleta ou divulgação do contexto. Por um lado, essa decisão de projeto favoreceu a flexibilidade no uso dos serviços, pois o desenvolvedor pode utilizar somente aqueles de que realmente necessita. Além disso, essa abordagem permite

que novos serviços possam ser incorporados à arquitetura de forma modular, tornando os serviços do núcleo da MoCA mais escaláveis por não sobrecarregá-los com a implementação das funcionalidades envolvidas na provisão dos mais diversos tipos de informação de contexto (e.g, contexto computacional, contexto pessoal). Por outro lado, essa decisão acarreta um forte acoplamento entre os serviços de provisão de contexto que precisam interagir uns com os outros para desempenhar suas funções. Por exemplo, o LIS usa no cálculo da inferência da localização as informações de RSSI providas pelo CIS, que, por sua vez, foram coletadas pelos Monitores executando nos dispositivos móveis dos usuários. Esse acoplamento configura uma dependência de execução entre os serviços. Ou seja, para utilizar o LIS, o CIS deve estar executando. Para auxiliar a implantação e uso desses serviços, é descrito em (3) um documento (*MoCA Overview*) que discute as questões de dependências de execução e implementação entre as APIs e serviços, e os passos necessários para desenvolver uma aplicação baseada na MoCA.

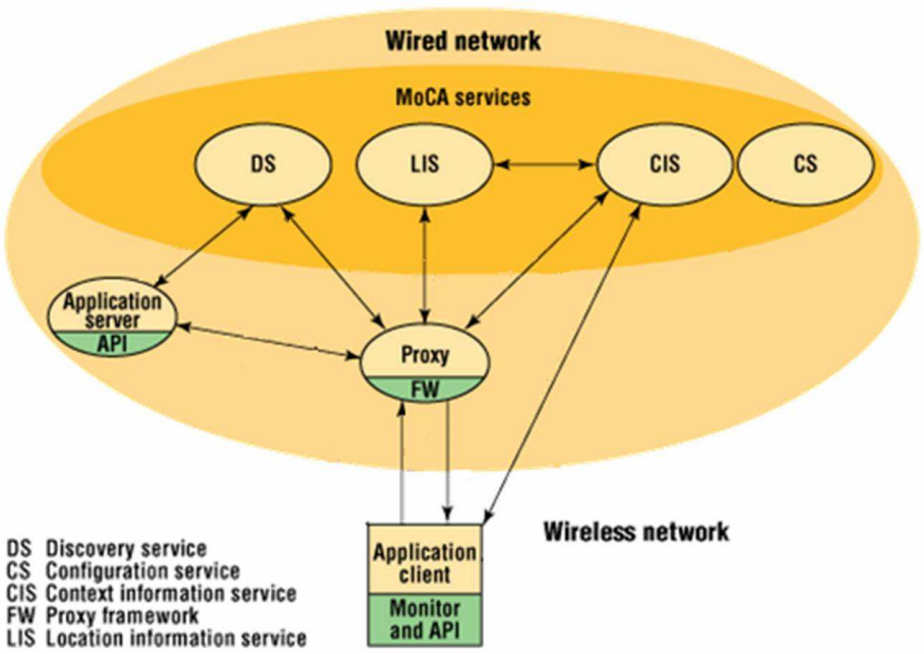

Figura 2.1: Arquitetura MoCA

Os serviços de provisão de contexto da MoCA são brevemente descritos nas subseções seguintes. Maiores detalhes sobre como esses interagem com os demais serviços da arquitetura podem ser encontrados em (3, 2). O CIS, o Monitor e as APIs de comunicação síncrona e assíncrona (baseadas em eventos) constituem parte da contribuição deste trabalho. O serviço de localização (LIS) que foi o principal elemento usado no estudo de caso para o controle de privacidade, no entanto, foi resultado de uma dissertação de mestrado (5).

Monitor

O Monitor coleta informações do dispositivo móvel e da rede sem fio e as envia periodicamente para o CIS na rede fixa. Tais informações (na forma de uma lista de variáveis) podem ser classificadas em duas categorias, como a seguir.

Informações coletadas da rede sem fio:

- intensidade do sinal e endereço MAC de todos os *Access Points* (APs) IEEE 802.11 que estão no raio de cobertura do dispositivo;
- endereço MAC e intensidade do sinal do AP corrente.

Informações coletadas sobre a configuração e os recursos disponíveis do dispositivo móvel:

- taxa de uso da CPU (em %);
- memória disponível (em Kbytes);
- endereço MAC e IP/Máscara;
- nível de energia disponível (em %).

Além dos envios periódicos das informações supracitadas, o Monitor também as enviará quando for detectada uma mudança no endereço IP ou *AP* corrente, já que isso caracteriza um *roaming* (*handover*) e, portanto, representa uma mudança do estado/contexto corrente do dispositivo que possivelmente é de interesse para as aplicações. As informações coletadas pelo Monitor podem ser utilizadas para diferentes propósitos, como, por exemplo, implementar uma adaptação dinâmica das aplicações em função do nível de energia ou da memória disponível.

Até o momento, temos uma implementação completa do Monitor para Windows XP (75) e Windows CE (76), e um protótipo para Linux. Para Windows XP, disponibilizamos duas versões, uma versão de produção que está integrada à MoCA e uma versão de demonstração que não depende dos serviços da arquitetura. A versão de demonstração pode ser utilizada para outros propósitos específicos, como, por exemplo, avaliar a qualidade do sinal dos *APs* da rede.

A maior dificuldade de implementação do Monitor foi o desenvolvimento do módulo responsável por obter as informações da rede sem fio no Windows XP. Para tanto, adaptamos e usamos o *driver* que implementa a arquitetura de comunicação com dispositivos de rede definida pela Microsoft, conhecida por *Network Driver Interface Specification* (NDIS) (77). Através deste, o Monitor

dispara e coleta os resultados dos *scans*[4] realizados na rede sem fio usando interfaces e comandos da *NDIS Wrapper*, que estão padronizados e são usados por *drivers* de rede de todos os fabricantes de placas IEEE 802.11. Uma visão geral da NDIS é ilustrada na Figura 2.2. Nessa figura, o Monitor age como uma aplicação que interage com o *NDIS Protocol Driver*, que customizamos a fim de permitir o envio de comandos (e.g., *SetScanning*, *GetScanningResult*) para o *NDIS Miniport Driver*, que é a implementação do driver da placa de rede de um dado fabricante.

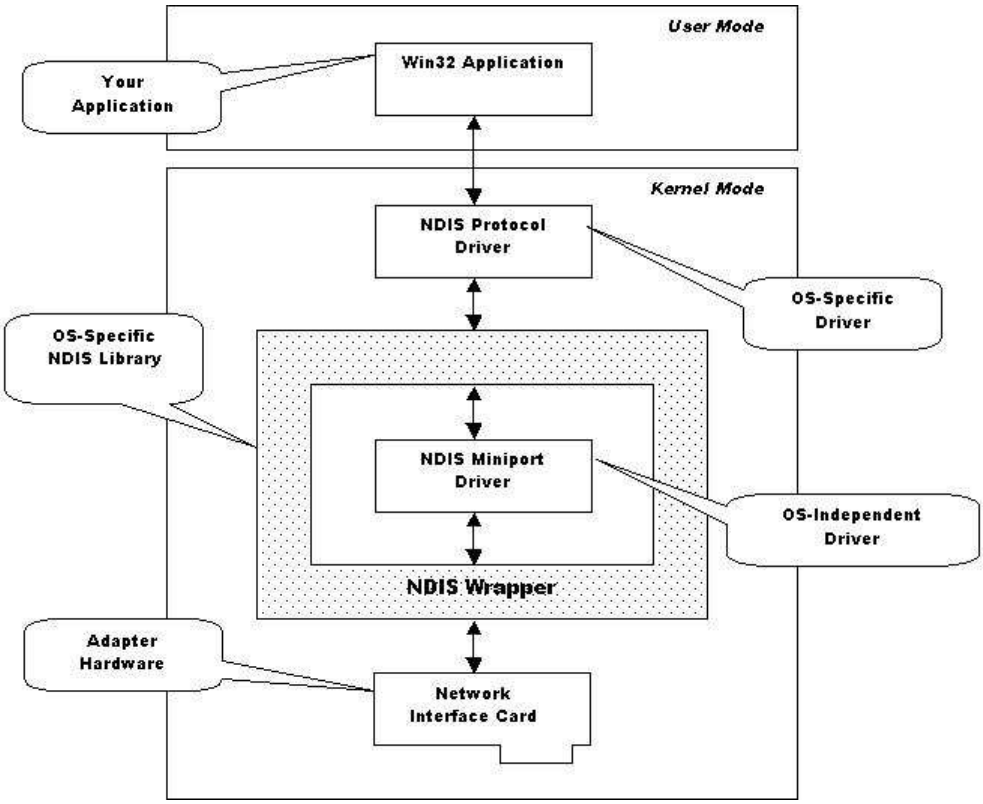

Figura 2.2: Arquitetura NDIS

Através da NDIS, a implementação do Monitor tornou-se independente do fabricante da placa de rede sem fio, pois toda a comunicação entre o Monitor e o driver da placa de rede ocorre via a *NDIS Wrapper*. As demais informações (CPU, nível de energia, ...) são obtidas utilizando APIs específicas definidas pela Microsoft. No Linux, a lista de APs com as respectivas intensidade do sinal é obtida através das bibliotecas disponibilizadas pela *Wireless Tools for Linux* em (78) e, as demais informações são coletadas a partir dos arquivos contidos no */proc* na estrutura de diretórios do Linux.

De uma forma geral, o serviço de sensoriamento é um componente chave para o desenvolvimento de qualquer arquitetura de provisão de contexto.

[4]Varredura na rede sem fio para obter a intensidade do sinal dos APs que estão no raio de cobertura do dispositivo.

No entanto, a implementação deste serviço geralmente demanda um grande esforço de programação por causa da *complexidade* em obter as informações acerca da utilização dos recursos de cada plataforma de *hardware* (e.g., iPAQs, Palms, Laptops), i.e., como obter a taxa de uso da CPU, bateria, ou, obter informações sobre cada possível interface de rede disponível. Como exemplo da complexidade envolvida no desenvolvimento deste serviço, vale destacar que, às vezes, a implementação da coleta de um contexto específico (e.g., uso da CPU ou intensidade de sinal dos APs 802.11) para dispositivos de uma mesma plataforma e de um mesmo fabricante, como, por exemplo, iPAQs da HP, pode variar de acordo com a série do produto ou versão do sistema operacional.

Serviço de informação de contexto

O *Context Information Service* (CIS) é um serviço que recebe, processa e divulga para as aplicações interessadas os dados de contexto recebidos de diversos monitores. Através das APIs de comunicação, *Communication Protocol* e *Event-based Communication Interface* (ECI), a aplicação pode interagir com o CIS de duas formas: por meio de uma comunicação síncrona (requisição/resposta) para obter o valor de uma dada variável de contexto ou por meio de uma comunicação assíncrona baseada em eventos para ser notificada sobre uma eventual mudança de valor de alguma variável de contexto. Através das notificações, as aplicações podem manter-se a par das mudanças do estado corrente do dispositivo móvel (e.g., memória livre, nível de energia) ou da rede sem fio (e.g., variação do RSSI, AP corrente).

Para ser notificada, a aplicação deve se registrar no CIS (através de uma *subscription*) passando como parâmetro um tópico ou assunto (Subject) e, opcionalmente, uma expressão (baseada no padrão SQL92) para filtrar os eventos de seu interesse. Através da API *ECI*, o CIS recebe e registra a *subscription* e, eventualmente, publica um evento que satisfaz as propriedades da assinatura de interesse de uma aplicação. Para o CIS, o tópico (i.e., o Subject) é o endereço MAC do dispositivo e as propriedades especificam uma expressão sobre o estado das variáveis de contexto deste dispositivo (e.g., *Roaming, FreeMemory*) nas quais a aplicação está interessada. Uma *subscription*, por exemplo, poderia ser *Subject* = {*"02:DA:20:3D:A1:2B"*}, *Properties* = {*"roaming = True" OR "FreeMem < 15%" OR "CPU > 90%" OR (("OnLine = False") AND ("DeltaT > 15s"))*}. A aplicação recebe então uma notificação sempre que a condição informada na expressão for satisfeita. Essa notificação descreve o valor corrente das *tags/variáveis* de contexto do dispositivo.

A aplicação também pode ser notificada quando a condição especificada

através da expressão voltar a ser invalidada i.e. a ter o valor "falso" (evento conhecido também como *negação da expressão*). Essa funcionalidade é essencial para as aplicações que implementam algum tipo de adaptação baseada em contexto. Por exemplo, após ser notificada que um determinado estado de alguma *tag* de contexto do dispositivo está abaixo do limiar aceitável (e.g., nível de energia abaixo de 10%), a aplicação pode desempenhar alguma função, que possivelmente tem alto custo computacional, para lidar com a situação corrente, como, por exemplo, ativar uma política de *cache*, compactar ou remover parte dos dados enviados para o cliente executando no dispositivo em questão. Por isto, a aplicação também deve ser notificada quando o referido estado do nível de energia voltar a ficar acima do limiar aceitável. Essa funcionalidade permite que a aplicação pare de executar a adaptação. As possíveis *tags* de contexto que podem ser utilizadas nas expressões de consulta ao CIS são descritas na Tabela 2.1.

Tag	**Valor**	**Descrição**
CPU	int	Uso da CPU (entre 0 e 100%)
EnergyLevel	int	Nível de energia disponível (entre 0 e 100%)
AdvertisementPeriodicity	int	Periodicidade em que o monitor envia suas notificações
APMacAddress	string	Endereço Físico (MAC) do Ponto de Acesso
RSSI	long	Intensidade do sinal para com o AP corrente (em .dBm)
FreeMemory	long	Total de memória disponível (em Kb)
DeltaT	long	Por quanto tempo o dispositivo está off-line (em milisegundos)
OnLine	boolean	Verdadeiro se o dispositivo estiver on-line (i.e com conectividade de rede)
IPChange	boolean	Verdadeiro se o dispositivo mudar de endereço IP
Roaming	boolean	Verdadeiro se o dispositivo implementar uma operação de roaming

Tabela 2.1: Atuais tags de contexto do CIS

Em linhas gerais, o CIS recebe as informações enviadas pelos Monitores, faz uma análise sintática no conteúdo da mensagem e extrai os dados que descrevem o contexto do dispositivo e da rede. Em seguida, ele publica tais dados, usando a API do serviço de eventos, para as aplicações que tenham se registrado como interessadas em alguma mudança de estado do dispositivo ou da rede. A implementação do CIS, da API ECI e *Communication Protocol* estão disponíveis para *download* em (79, 73, 80).

Serviço de inferência de localização

O *Location Inference Service* (LIS) infere e disponibiliza a localização simbólica dos dispositivos móveis em áreas cobertas por *Access Points (APs)* de

redes IEEE 802.11. Para isso, ele utiliza a intensidade de sinais RSSI coletados e divulgados pelos monitores e CIS, respectivamente.

Conforme descrito em (5), a inferência é feita a partir da comparação da similaridade entre o padrão de sinal de RF atual (representado por um vetor, onde cada elemento deste vetor corresponde ao sinal de um AP "audível" ao dispositivo) e o padrão (vetor RSSI) medido anteriormente em pontos pré-definidos em uma região com cobertura de rede IEEE 802.11. Portanto, em uma primeira etapa (fase de calibração) as amostras de vetores RSSI são coletadas em *pontos de referência* bem definidos na(s) área(s) de interesse, e armazenadas em um banco de dados do LIS. Considerando que os sinais de rádio são suscetíveis à intensa variação e interferência, a precisão da inferência da localização depende fortemente do número de APs, do número de pontos de referência escolhidos e do número de amostras de vetores RSSI coletados.

O LIS atende, principalmente, aplicações que necessitam conhecer a posição de um dispositivo em termos de regiões simbólicas (ao invés de coordenadas), onde tais regiões são áreas geográficas não menores do que 1 a $4m^2$. Devido a flutuação intrínseca no sinal de rádio, a inferência da localização baseada em RSSI não é capaz de oferecer resultados com maior precisão. Entretanto, acreditamos que a precisão da localização obtida pelo LIS é suficiente para uma grande classe de aplicações sensíveis à localização.

Ao invés de coordenadas físicas (e.g., latitude/longitude), várias aplicações sensíveis à localização necessitam apenas saber a localização simbólica de usuários e dispositivos. No LIS, regiões simbólicas podem ser estruturadas hierarquicamente e são representadas como qualquer área geográfica com uma semântica (ou identificação) bem definida, tal como uma sala específica, corredor, andar de um prédio, prédio, etc. Essa funcionalidade permite que as aplicações LBS obtenham a informação de localização em diferentes granularidade. Para tanto, o LIS implementa um modelo hierárquico de localizações, como aqueles discutidos nos trabalhos (65, 81). Neste modelo (naturalmente representado como uma árvore), regiões simbólicas podem ser compostas por outras regiões simbólicas menores (aninhadas). Nesta hierarquia, os nós folhas são definidos como regiões simbólicas *atômicas*, as menores unidades de localização que podem ser reconhecidas pelo LIS. Os ancestrais de um nó folha da hierarquia de regiões representam as demais regiões simbólicas as quais um dado dispositivo pode estar implicitamente associado. Por exemplo, dada a hierarquia "PUC-Rio/Prédio RDC/5º Andar/Sala 501", o LIS pode inferir que um dado dispositivo se encontra no "Prédio RDC" caso a localização corrente desse dispositivo seja "Sala 501". O LIS oferece uma interface através da qual cada aplicação pode criar e gerenciar sua própria topologia de regiões simbólicas

baseada nas regiões atômicas definidas pelo administrador do serviço.

2.5 Discussões

Apesar de existirem várias outras infra-estruturas de provisão de contexto (65, 81, 66, 67, 68, 69, 70), que inferem e divulgam a localização do usuário para redes IEEE 802.11, nós decidimos projetar e implementar uma infra-estrutura equivalente por uma série de razões tais como: o interesse do grupo em adquirir experiência no desenvolvimento de serviços de provisão de contexto; a especificidade da maioria das arquiteturas proposta (como esperado, cada uma delas atende a objetivos específicos dos projetos nos quais foram desenvolvidas); a dificuldade em manter um *software* paralelamente desenvolvido por outro grupo; a incompatibilidade tecnológica em termos de linguagem e ambiente de desenvolvimento, a falta de documentação e restrições de direitos autorais para alterar e disponibilizar o código fonte.

A partir do desenvolvimento da MoCA, foi possível testar e avaliar o quanto uma infra-estrutura de provisão de contexto auxilia o processo de desenvolvimento de aplicações sensíveis ao contexto. Em particular, as funcionalidades de coleta implementadas pelo Monitor, de processamento e divulgação das informações de contexto implementadas pelo CIS e LIS oferecem ao desenvolvedor uma abstração significativa que facilita e agiliza o desenvolvimento dessas aplicações. Além desses serviços, as APIs de comunicação síncrona e baseada em eventos oferecem ao desenvolvedor uma maior flexibilidade de consulta ao contexto requerido pela aplicação. Essas permitem a aplicação consultar o valor corrente de uma variável de contexto ou ser notificada quando ocorrer um determinado evento de seu interesse (e.g., ser notificada quando um usuário entrar ou sair em uma determinada sala ou quando o nível de bateria do dispositivo estiver abaixo do limiar requerido).

O projeto da arquitetura MoCA também serviu de base para o desenvolvimento de várias outras pesquisas de mestrado e de doutorado realizadas no laboratório LAC e em outros grupos de pesquisa (82, 83). Geralmente, esses trabalhos usam diretamente o contexto divulgado pelos serviços da MoCA para propor inovações de pesquisa relacionadas ao desenvolvimento ou ao uso de aplicações sensíveis ao contexto. Por exemplo, o ProxyFramework (74) oferece um *framework* para o desenvolvimento de *proxies* sensíveis ao contexto que implementam adaptações baseadas em conteúdo de acordo com o contexto corrente dos dispositivos móveis e da rede sem fio. Outros trabalhos usam as informações disponibilizadas pelo CIS para inferir novas informações de contexto. Seguindo essa idéia de trabalhos cooperativos/complementares, a dissertação

(84) propõe um *framework* chamado *FLoCS (Framework for Location-based Communication Services)* que auxilia o desenvolvimento de aplicações LBS e, a dissertação publicada em (85) propõe um *Serviço de Matching de Perfil* que utiliza o LIS para identificar usuários co-localizados cujos interesses declarados em seus perfis são similares.

Dentre os serviços da infra-estrutura de coleta e distribuição de contexto, o Monitor foi implementado em ANSI C e as APIs de comunicação e o CIS em Java somando aproximadamente 30.000 linhas de código. O desenvolvimento dessa infra-estrutura foi essencial para avançarmos na pesquisa de privacidade, pois essa auxiliou o desenvolvimento do LIS e de algumas aplicações LBS tais como *NITA, UGuide, Buddy Space Live, Wireless Marketing Service*, dentre outras (4). O uso destas nos motivou a tratar das questões de privacidade relacionadas à informação de localização. Pois, embora estas aplicações ofereçam vários serviços (e.g., localizar um usuário, enviar uma mensagem para os usuários que passarem por uma determinada região, etc.), elas podem também ser usadas para derivar conclusões de onde e como o usuário está usando o seu dispositivo.

Neste trabalho, nós focamos nas questões de privacidade associadas à informação de localização pelos seguintes motivos: para estreitar o nosso foco de pesquisa, pelo fato da MoCA fornecer esse tipo de informação e, principalmente, porque a informação de localização evidencia melhor as necessidades de privacidade dos usuários em relação ao uso de aplicações LBS.

No caso das informações coletadas pelo Monitor e divulgadas pelo CIS, há vários níveis de riscos envolvidos: a localização do usuário poderia ser inferida em uma granularidade mais grossa a partir do endereço IP, do endereço MAC do AP corrente ou da lista de APs no raio de cobertura do dispositivo. A grande dificuldade em tratar das questões de privacidade relacionadas a essas informações está na complexidade de projetar um serviço que ofereça ao usuário final um mecanismo de controle de acesso único que gerencie as tentativas de acesso a *todas* as informações (e.g., endereço IP, MAC do dispositivo, identificação do crachá, etc) que podem, de alguma forma, revelar a sua localização. Ao invés de investigar uma abordagem mais genérica para lidar com tais questões, nós nos concentramos em tratar do controle de privacidade associado mais diretamente à informação de localização inferida pelo serviço de contexto (e.g., LIS), não tratando dos casos em que a localização do usuário é divulgada pelo endereço IP ou endereço MAC do seu dispositivo registrado no *log* de acesso de algum serviço. No Capítulo 3, nós descrevemos um modelo conceitual e os requisitos de privacidade que delinearam o projeto e implementação do serviço de privacidade proposto.