ARQUITETURA CIENTE DE CONTEXTO PARA APLICAÇÕES SOCIAIS MÓVEIS

RAFAEL GUIMARÃES SIQUEIRA

ARQUITETURA CIENTE DE CONTEXTO PARA APLICAÇÕES SOCIAIS MÓVEIS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Antônio Alfredo Ferreira Loureiro

Belo Horizonte

Maio de 2012

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Siqueira, Rafael Guimarães

S618a Arquitetura ciente de contexto para aplicações sociais móveis / Rafael Guimarães Siqueira — Belo Horizonte, 2012.
xviii, 70 f.: il.; 29 cm.

Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Antônio Alfredo Ferreira Loureiro

1. Computação - Teses. 2. Arquitetura de redes de computador - Teses. 3. Sistemas de comunicação móvel - Teses. 4. Middleware – Teses. I. Orientador. II. Título

519.6*21(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Arquitetura ciente de contexto para aplicações sociais móveis

RAFAEL GUIMARÃES SIQUEIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. ANTONIO ALFREDO FERREIRA LOUREIRO - Orientador
Departamento de Ciência da Computação - UFMG

DR. PEDRO OLMO STANCIOLI VAZ DE MELO
Bolsista Pós-Doutorado DCC - UFMG

PROFA. RAQUEL APARECIDA DE FREITAS MINI
Instituto de Ciências Exatas e Informática - PUC-MG

PROF. RICARDO AUGUSTO RABELO OLIVEIRA
Departamento de Computação - UFOP

Belo Horizonte, 30 de maio de 2012.

Resumo

Aplicações sociais móveis exploram o potencial das redes sociais através das plataformas móveis. Elas representam aplicações *peer-to-peer*, tanto pelo seu caráter social quanto pela pessoalidade dos dispositivos móveis. No entanto, em muitos casos, ainda são implementadas utilizando o modelo cliente-servidor. Na prática, isso significa que a comunicação entre dois dispositivos próximos fisicamente acontece através de servidores localizados, muitas vezes, em outros continentes, resultando em tempos maiores de resposta. Nos casos pontuais de implementações segundo o modelo *peer-to-peer*, são utilizados protocolos de comunicação inadequados aos ambientes nos quais o conteúdo dos nós está em constante mudança, como é o caso do contexto dos usuários nos ambientes de computação móvel. Para a implementação de aplicações sociais utilizadas a partir desses dispositivos, dentre eles os *smartphones* e *tablets*, é ideal que a infraestrutura utilizada seja criada, desde a concepção, para suportar a comunicação em redes sociais móveis. Este trabalho apresenta uma arquitetura que provê essa infraestrutura. Ele propõe a criação de uma camada *overlay peer-to-peer* ciente de contexto que agrupa usuários, cuja probabilidade de interação na rede social é maior. Os fatores que, potencialmente, incrementam essa probabilidade são a pré-existência de um relacionamento explicitamente definido ou a coincidência de características que possibilitam o estabelecimento desse relacionamento no futuro. O protocolo proposto nessa arquitetura utiliza descrições do conhecimento e dos contextos de usuários através de predicados lógicos, a partir dos quais é possível inferir contextos mais complexos e mais úteis do ponto de vista das aplicações. Essa abordagem é particularmente flexível por permitir a representação do conhecimento relevante para diferentes domínios, na forma desses predicados lógicos. A escalabilidade dos algoritmos é testada através de experimentos em um simulador discreto de eventos, que permite também determinar, empiricamente, características da arquitetura como o custo imposto pelo protocolo a cada um dos nós. Além disso, a validade da solução proposta é demonstrada através de uma aplicação social para a convivência no campus de uma universidade, testada utilizando conjuntos de dados reais representando a mobilidade de usuários em uma

universidade. A arquitetura implementada neste trabalho contribui para o desenvolvimento de aplicações sociais móveis mais condizentes com o paradigma da computação ubíqua, pois propõe uma forma de organização para a infraestrutura mais adequada a essas aplicações e um protocolo ciente de contexto que permite organizá-las dessa forma.

Palavras-chave: Aplicações Sociais Móveis, Computação Ubíqua, Middleware.

Abstract

Mobile social applications explore the potential of social networks on mobile platforms. They represent peer-to-peer applications, for their social aspect and the individuality of mobile devices. Nonetheless, in many cases, they are still deployed using a client-server model. In practice, this means that communications between two devices physically collocated happens through servers located in other continents, resulting in higher response times. On isolated cases in which these applications are deployed using peer-to-peer models, they use communication protocols inappropriate for environments where node content is constantly changing, such as user contexts on mobile computing environments. To implement social applications for these mobile devices, amongst them smartphones and tablets, it is ideal that their underlying infrastructure is created, from its conception, to support communications on mobile social networks. This work presents an architecture that provides such infrastructure. It proposes the creation of a context-aware peer-to-peer overlay that clusterizes users, whose probability of interaction on the social network is higher. Factors which, potentially, increment this probability are the pre-existence of an explicitly defined relationship or the coincidence of characteristics which could catalyze the establishment of this relationship in the future. The proposed protocol in this architecture uses descriptions of knowledge and user contexts as logic predicates, from which it is possible to infer other contexts more useful from the application's point of view. This approach is particularly flexible because it allows for the representation of relevant knowledge from many different problems, by means of these logic predicates. Algorithm scalability is tested with experiments on a discrete event simulator, which allows for empirical analysis of impact on individual nodes. Furthermore, the viability of the solution is demonstrated by an example mobile social application for campus community living, tested using a dataset with mobility traces of campus inhabitants. The architecture implemented in this work contributes for the development of mobile social applications more adequate to the ubiquitous computing paradigm, by providing a self-organizing infrastructure more suited for these applications and a context-aware protocol that organizes them in that

manner.

Keywords: Mobile Social Applications, Ubiquitous Computing, Middleware.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Sumário

Capítulo 1

Introdução

Inicialmente, as aplicações sociais móveis eram concebidas para explorar o potencial das redes formadas pela conexão oportunista entre usuários, possibilitada por sua proximidade física momentânea [Ioannidis et al., 2009; Mtibaa et al., 2008; Chaintreau et al., 2008]. Porém, com a massificação do uso dos *smartphones* e a adoção de tecnologias de comunicação de dados através das redes celulares, essas aplicações se livraram dos requisitos de proximidade física para a comunicação, embora em alguns casos isso ainda seja usado na lógica das aplicações e, consequentemente, se diversificaram através do uso em diversos domínios. Exemplos dessas aplicações envolvem o uso corporativo, através de ferramentas de colaboração [Schuster et al., 2010; Jansen, 2011], uso médico para monitoramento de pessoas carentes de cuidados especiais [Chang, 2008] e educacional [Khaddage & Lattemann, 2009].

Para alguns pesquisadores, um *smartphone* deve possuir um sistema operacional completo e uma plataforma de desenvolvimento [Greenhalgh, 2007], enquanto para outros é um celular com funcionalidades avançadas [Nolan, 2011; Allam, 2009]. Entretanto, sua pessoalidade e omnipresença o credenciam como a melhor ferramenta para acesso às aplicações sociais móveis. Além disso, através de seus sensores, ou quando combinados com soluções de computação vestível (*wearable computing*) [Starner et al., 1997; Starner, 1996], os *smatphones* permitem a coleta de informação distribuída, cuja interpretação pode detectar situações ou estados coletivos de interesse público ou particular. Um exemplo disso é uma aplicação distribuída de percepção de desastres (*disaster assessment*). Nessa aplicação, sensores de aceleração e bússolas, presentes em grande parte dos *smartphones* atuais, coletam o contexto de cada usuário individualmente e, quando essas informações são correlacionadas, permitem detectar a ocorrência de um terremoto [Lien et al., 2009], uma mudança no centro de gravidade da Terra, no seu norte magnético ou algum outro contexto coletivo de interesse.

A fim de que sejam criadas aplicações sociais mais adequadas aos dispositivos móveis, é necessário estabelecer melhores infraestruturas de comunicação. Atualmente, observa-se que a comunicação através de aplicações sociais móveis é implementada, na maior parte dos casos, utilizando servidores como mediadores, apesar de ser tipicamente uma comunicação direta entre dois ou mais usuários móveis. Embora existam benefícios no uso desse paradigma de comunicação, dentre eles a simplicidade das implementações e a diminuição do processamento nos dispositivos móveis, é possível obter maior flexibilidade na topologia se utilizada uma camada *overlay peer-to-peer*, i.e., uma camada de rede sobreposta à infraestrutura original, na qual cada participante atua como um agente autônomo, tomando decisões individuais e em prol do bem coletivo. Essa flexibilidade é desejável por permitir adaptar as aplicações às alterações no contexto, que é a representação das expectativas, preferências, características e estados do usuário e seu ambiente [Satyanarayanan, 2001; Davies & Gellersen, 2002], através da modificação da topologia da rede. Uma camada intermediária de software que controla esses aspectos sem onerar o desenvolvedor das aplicações contribui para a obtenção de aplicações sociais mais adaptadas aos ambientes móveis e ainda direciona o foco dos desenvolvedores para as funcionalidades das aplicações e a correlação de dados em aplicações distribuídas.

O presente trabalho apresenta uma arquitetura que representa exatamente essa camada intermediária e provê a infraestrutura necessária para o funcionamento das aplicações sociais móveis, segundo as peculiaridades de cada domínio. Uma arquitetura de software envolve a descrição dos elementos que compõem a aplicação e a interação entre eles, padrões que direcionam sua composição e as restrições nesses padrões [Shaw & Garlan, 1996]. A arquitetura proposta neste trabalho propõe a organização dos componentes das aplicações sociais de uma forma que beneficia o seu uso a partir de dispositivos móveis. O cerne da arquitetura é uma camada *overlay peer-to-peer* ciente de contexto, que proporciona a formação de grupos de usuários com características semelhantes. A ciência de contexto é particularmente relacionada ao contexto social do usuário, que é o conjunto de características que influenciam no comportamento social do usuário em termos da criação de novos relacionamentos ou a interação através de outros pré-estabelecidos. Através da intervenção dos líderes desses clusters é possível realizar buscas por usuários que possuam contextos semelhantes àquele procurado e monitorar alterações em seus contextos, com o intuito de informar alguma parte interessada. Além disso, foi concebida uma aplicação social móvel, chamada de *Social Campi*, que ilustra a utilidade da arquitetura proposta.

Uma análise da literatura (detalhada no capítulo 2) revela a existência de diversos trabalhos que desenvolvem camadas intermediárias de software ou middleware para a

composição de aplicações móveis e outras que lidam com a infraestrutura necessária para aplicações sociais. No entanto, não foi encontrada uma arquitetura projetada para suportar o desenvolvimento de aplicações sociais móveis e os trabalhos encontrados que implementam esse tipo de arquitetura o fazem para suportar alguma aplicação social móvel específica e não proporcionam uma arquitetura reusável para diferentes tipos de aplicações.

1.1 Motivação

Projeções indicam que por volta do ano de 2014, a Apple alcançará o impressionante número de 100 bilhões de aplicativos vendidos na sua loja virtual [Jackson, 2011]. Para compreender a grandeza desse número, isso significa que em 2014 ela ultrapassará o número de sanduíches *Big Mac* vendidos historiaicamente pela rede *McDonald's*, uma das maiores cadeias de *fast food* do mundo. Há projeções parecidas em volume de vendas para aplicativos da plataforma Android, o sistema operacional para dispositivos móveis da Google [Jackson, 2011]. No entanto, a diferença entre as linguagens de programação, arcabouços e ambientes integrados de desenvolvimento, dificulta a criação de soluções de software que funcionem simultaneamente nessas várias plataformas. Gradualmente surgem ferramentas para desenvolvimento multiplataforma [Christ, 2011; Allen et al., 2010; Oehlman & Blanc, 2011] para auxiliar no desenvolvimento de aplicativos para dispositivos móveis, porém, seu foco é na geração automática de código através de interpretadores léxicos. Portanto, ao criar uma aplicação social móvel, o desenvolvedor, além de lidar com questões relacionadas às funcionalidades da aplicação, precisa lidar com problemas possivelmente fora da alçada de seu conhecimento, dentre eles a implementação da infraestrutura necessária ao funcionamento de seu aplicativo. Adicionalmente, as peculiaridades dos dispositivos móveis, quando exploradas corretamente, permitem aplicações improváveis nas plataformas de computadores pessoais, como explicitado a seguir.

Suponha o seguinte cenário: no campus de uma universidade existem diversos usuários de dispositivos móveis, dentre eles celulares, *tablets* e computadores portáteis. A administração da universidade planeja uma aplicação social que proporcione meios de comunicação para docentes e discentes através de avisos e comunicados. Ela também quer proporcionar mecanismos para a comunicação entre os professores e alunos sem distinção. Além disso, é desejável que a comunicação feita através da aplicação não perturbe o andamento das aulas. Assim, deve haver funcionalidades que permitam adaptar as notificações ao estado atual dos usuários. A aplicação descrita se trata de

uma aplicação social móvel, pois proporciona formas de interação social, nas quais o contexto dos usuários é avaliado e determinante para sua ocorrência. Embora essa aplicação seja propositalmente simples, existem requisitos não-funcionais advindos de peculiaridades desse tipo de aplicação. Primeiramente, por se tratar de uma aplicação ciente de contexto, será necessário definir o conjunto de entidades, seus atributos e relacionamentos que irão compor o contexto dos usuários. Em seguida, será necessário implementar formas de coletar os atributos dessas entidades e ainda um mecanismo de inferência de contextos, capaz de racionalizar a partir desses atributos e alcançar conclusões em nível mais alto e mais relevante para as aplicações. Como a interação através da aplicação tem caráter social, existe uma tendência à formação de grupos cuja probabilidade de interação é maior, como explicado na seção 2.2. Portanto, é desejável que a topologia, com a qual se organizam os participantes da aplicação, seja capaz de antecipar essa tendência de clusterização, reduzindo a distância entre usuários que se comunicam com maior frequência. Será necessário ainda estabelecer um protocolo que permita a comunicação entre clientes móveis e possua funcionalidades de *publish/subscribe* (pub/sub) para monitorar mudanças no contexto dos usuários. O desenvolvedor deve implementar as funcionalidades desejadas para a aplicação através da correlação de informações de contexto e da atuação no ambiente através do envio de notificações. Além disso, é importante que a aplicação considere os recursos dos dispositivos, como energia e largura de banda, a privacidade e preferências de interação dos usuários com a aplicação. A partir desse cenário, é possível designar os seguintes componentes para uma aplicação social móvel, expressos também na figura 1.1:

1. Mecanismo de coleta e inferência de contextos;
2. Protocolo de comunicação pub/sub;
3. Mecanismo de correlação de dados e atuação no ambiente;
4. Camada *overlay peer-to-peer*;

Para o primeiro componente, existem linguagens de descrição de contexto baseadas em ontologias, dentre elas a OWL [McGuinness et al., 2004] e a OWL2 [Motik et al., 2009](*Web Ontology Language*), iniciativas do W3C (*World Wide Web Consortium*) que têm como objetivo padronizar ontologias para descrição de entidades na Internet. Além disso, existem ferramentas de inferência de predicados lógicos a partir de ontologias, desenvolvidas especificamente para inferência de contextos para aplicações móveis [Kofod-Petersen & Mikalsen, 2005; Strang et al., 2003; Preuveneers et al., 2004]. Uma alternativa interessante às ontologias é o uso de modelos lógicos, descritos

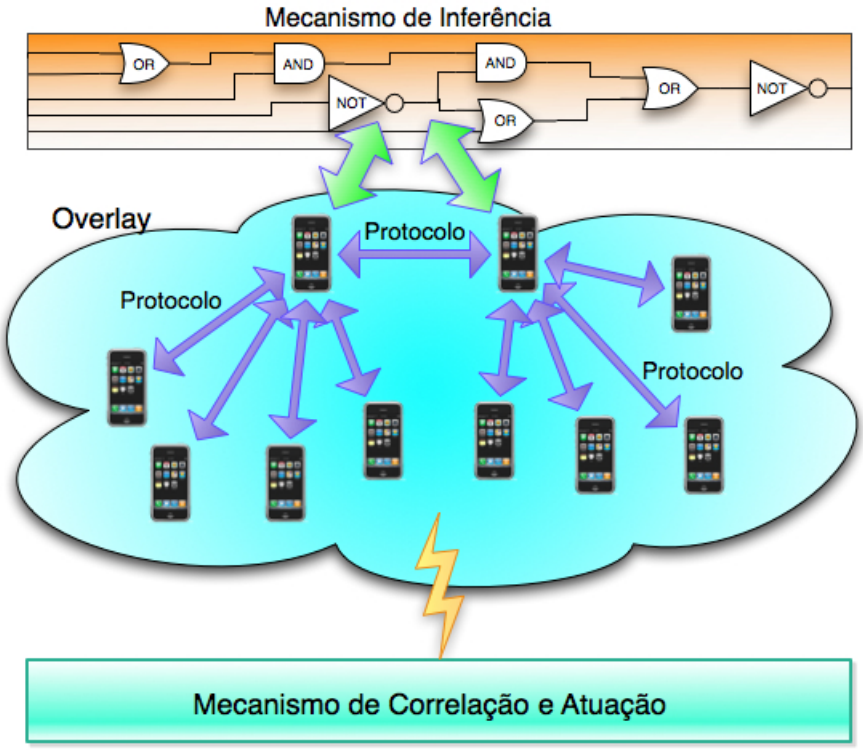


Figura 1.1. Componentes de uma aplicação social móvel

em linguagens funcionais, como por exemplo o Prolog ou Haskell, para descrever o contexto dos usuários, que poderá ser analisado pelos próprios interpretadores dessas linguagens.

Na literatura, existem alguns protocolos de comunicação com suporte a pub/sub. Em Lubke et al. [2011], os autores utilizam o protocolo XMPP na formação de grupos para publicação e consumo de conteúdo, baseados na proximidade física entre os usuários de redes sociais móveis. Uma estratégia diferente é usada em Triantafillou & Aekaterinidis [2004]. Nesse trabalho os autores implementam um protocolo para gerenciamento da topologia de uma rede p2p estruturada e, ao invés da formação de grupos, manipulam a topologia para beneficiar a publicação e o consumo de conteúdo. Embora seja possível utilizar uma dessas implementações para o desenvolvimento de aplicações sociais móveis, o protocolo escolhido influencia diretamente na topologia da rede p2p formada pelos usuários da aplicação e, no caso das aplicações sociais móveis, é interessante que essa topologia reflita os relacionamentos sociais entre os usuários.

O terceiro componente também é tratado em diversos trabalhos na literatura, embora esses mecanismos sejam específicos aos problemas para os quais foram desenvolvidos, dificultando o seu reúso. A correlação de informações de contexto é usada

em [Davis et al., 2004] para a geração de metadados para descrever fotografias a partir de informações espaciais, temporais e sociais. Nardi [1996] explica o papel da teoria das atividades com relação à interação humano-computador. A detecção de atividades é comumente implementada através da coleta e correlação de contextos e o desenvolvimento de aplicações mais interativas a partir desses contextos é uma mecanismo de atuação muito utilizado.

Finalmente, existem exemplos na literatura de camadas *overlay* P2P para aplicações sociais móveis. Tsai et al. [2009] apresentam um modelo de infraestrutura para esse tipo de aplicação baseado no protocolo P2P Juxtapose (JXTA), que proporciona a comunicação através de mensagens XML e independente de topologia. Mani et al. [2009] apresentam um sistema para criação e implantação de redes P2P espontâneas, com suporte para dispositivos móveis. Entretanto, não se tem conhecimento de nenhum trabalho que implemente uma camada *overlay* reusável cuja topologia seja adaptável ao contexto social dos usuários e que otimize as comunicações em aplicações sociais móveis. Além disso, é desejável que essa camada *overlay* interaja de forma modular com mecanismos de inferência de contextos, correlação de dados e atuação no ambiente, formando uma arquitetura para aplicações sociais móveis que contribua para o desenvolvimento de aplicações sociais mais adaptadas ao paradigma da computação ubíqua.

1.2 Objetivos

Este trabalho trata do desenvolvimento de aplicações sociais móveis com uma característica em comum: usuários com contextos similares têm maior probabilidade de se relacionarem. Essa característica não é comum a todas as aplicações ou redes sociais. Vaz de Melo [2011] demonstra que em uma rede social formada por usuários de táxis em San Francisco, o contato entre usuários é aleatório. Porém, a prática e a pesquisa mostram que é possível verificar um maior número de conexões entre indivíduos de contexto semelhante, na maioria das aplicações, quando os fatores motivantes para os relacionamentos sociais estão expressos no contexto [Backstrom et al., 2006; Levine & Kurzban, 2006; Wang et al., 2004]. Esses fatores motivantes constituem o contexto social do usuário, que engloba toda a informação relevante para a caracterização de uma situação que influencie a interação de um usuário com outros usuários [Wang et al., 2004]. Um exemplo de informação, que tem influência na probabilidade de comunicação entre dois usuários, é a localização geográfica [Liben-Nowell et al., 2005; Adamic & Adar, 2005].

O objetivo deste trabalho é propor uma arquitetura para aplicações sociais móveis com uma topologia adaptável ao contexto social dos usuários. Essa topologia flexível é proporcionada por uma camada *overlay* P2P e um protocolo ciente de contexto. Aspectos relacionados à inferência e comparação de contextos são agregados de forma modular, através de um interpretador lógico implementado em Prolog. A interface para interação com as aplicações possui um método de busca por contexto e um método de *pub/sub* para o monitoramento de contextos, além dos métodos de adesão e saída necessários para o funcionamento da camada *overlay*.

1.3 Contribuição do Trabalho

Uma análise detalhada da literatura (capítulo 3) revela que existem arcabouços, arquiteturas e middlewares para o desenvolvimento de aplicações sociais e aplicações móveis. No entanto, poucos desses trabalhos têm o foco na interseção desses dois tipos de aplicações. Dentre estes, alguns lidam com desafios associados à conexão oportunista [Pietiläinen et al., 2009; Chaintreau et al., 2008], outros com aspectos pontuais de aplicações sociais, como a formação de grupos [Bottazzi et al., 2006], e de aplicações móveis, como a coleta de contexto [Gupta et al., 2009]. O trabalho mais semelhante encontrado na literatura é [Mokhtar et al., 2009], no qual os autores implementam um middleware para a recomendação de relacionamentos para nós participantes de uma camada *overlay peer-to-peer*, com base na proximidade física e social entre dois usuários que desempenham uma mesma atividade. Não se tem conhecimento, entretanto, de arquiteturas para desenvolvimento de aplicações sociais móveis que apresentem estes dois algoritmos de interesse científico:

1. Construção de infraestruturas cientes de contexto para redes sociais móveis que trate do problema da auto-organização da rede P2P em função de similaridades de contexto, e da reorganização da rede no caso de ocorrência de falhas ou desligamento de nós;

2. *Publish/subscribe* para alterações no contexto implementado através da intervenção de líderes de cluster.

O algoritmo de construção de infraestruturas ciente de contexto é particularmente interessante para as redes sociais, pois sua abordagem é baseada na clusterização, que é uma tendência natural das redes sociais, quando sua formação não é aleatória. Os algoritmos foram avaliados através de uma aplicação social de convivência no campus de uma universidade, que demonstra a aplicabilidade do mecanismo *publish/subscribe* e da

clusterização da rede. Além disso, a escalabilidade dos algoritmos foi testada através de experimentos com modelos lógicos criados aleatoriamente em Prolog, simulando a representação de problemas em diversos domínios. A partir desses experimentos foi possível determinar parâmetros necessários para o funcionamento da arquitetura, como o número ideal de clusters em função do número total de participantes e da diversidade de contextos possíveis.

1.4 Organização do Texto

A solução proposta para o problema do desenvolvimento de aplicações sociais móveis é uma arquitetura de agentes autônomos organizados em uma camada *overlay peer-to-peer* e clusterizados de forma ciente de contexto. Essa clusterização considera também o aspecto social de cada usuário, sua tendência a se comunicar em sua rede social e a criar novos relacionamentos. Para compreender a solução enunciada, são apresentados conceitos básicos no capítulo 2. O capítulo 3 compara a arquitetura proposta com as pesquisas relacionadas, ressaltando suas qualidades e limitações. O capítulo 4 apresenta os detalhes da arquitetura proposta, chamada CAWTEC, e a metodologia necessária para o desenvolvimento de aplicações sociais móveis utilizando esta arquitetura. O capítulo 5 demonstra a validade da solução através de experimentos de simulação e explica o desenvolvimento de uma aplicação concebida para ilustrar a viabilidade da arquitetura. Finalmente, o capítulo 6 apresenta as conclusões e uma avaliação dos trabalhos futuros que podem contribuir para o desenvolvimento de aplicações sociais móveis.

Capítulo 2

Referencial Teórico

Em 1991, Mark Weiser publicou um relato pessoal a respeito do que imaginava para a computação do século 21 [Weiser, 1991]. Àquela época seus esforços de pesquisa e de sua equipe já almejavam a criação de espaços inteligentes, que seriam possibilitados por dispositivos que eles já utilizavam em 1991, dentre eles um dispositivo sensível ao toque, semelhante aos *tablets* atuais, chamado *Pad*, uma espécie de *smartphone* chamado *Tab* e um tipo de quadro de avisos interativo, chamado *Board*. Seu trabalho estabeleceu a base da computação ubíqua. Segundo ele, as melhores tecnologias são aquelas que desaparecem no cotidiano das pessoas até que estas não possam distinguí-las em seu ambiente. Portanto, a computação ubíqua almeja a transparência na interação do usuário com os dispositivos que integram seu ambiente.

Durante algum tempo, as pesquisas na área da computação ubíqua se concentraram nos ambientes inteligentes projetados para interagir dessa forma com os usuários. Embora essas pesquisas sejam restritas a ambientes controlados, diferentes das redes sociais móveis, a partir delas surgiu o conceito de contexto de usuário. Esse contexto é o aspecto central da arquitetura proposta neste trabalho e está explicado em detalhes neste capítulo. Além disso, para compreender bem o funcionamento da arquitetura, é necessário compreender o que é uma camada *overlay peer-to-peer*, o que são clusters – forma de organização da arquitetura, o que são agentes autônomos – maneira que se comportam os participantes, e o que são as redes sociais e suas características principais. Todos esses conceitos são apresentados neste capítulo.

2.1 Contexto

Existem diferentes definições para o contexto, justificadas por seu uso diverso. Uma apropriada a este trabalho é a de Chen & Kotz [2000], que define o contexto como

um conjunto de estados do ambiente e configurações que determinam o comportamento de uma aplicação ou nos quais um evento da aplicação, de interesse do usuário, ocorre. Esse conceito é compartilhado pela maioria das aplicações com preocupação em adaptar-se às características do ambiente de execução e do dispositivo e preferências ou estado do usuário. Por essa razão, existem diversas formas propostas para representar o contexto, variando em função de sua capacidade representativa, compatibilidade com diversos dispositivos, aplicabilidade e qualidade de informação. É possível classificar as abordagens para a representação do contexto da seguinte forma [Chen & Kotz, 2000; Strang & Linnhoff-Popien, 2004]:

- Chaves-Valores: nesse modelo, o contexto é descrito através de uma lista de atributos e os algoritmos de descoberta realizam uma busca pelo nó que possua os atributos desejados. Os principais exemplos dessa abordagem estão descritos em Schilit [1995], que apresenta uma arquitetura para computação ciente de contexto, na qual o contexto é armazenado usando a localização como chave para os atributos de entidades de contexto e Samulowitz et al. [2002], que embora utilize o conceito de entidades abstratas para se referir às entidades do contexto, as representa como um conjunto de atributos cuja chave é o nome da entidade.

- Linguagens de Marcação (*Markup Languages*): conteúdo definido hierarquicamente e recursivamente usando *tags* de marcação; Dey [2001], Capra et al. [2001] e Kagal et al. [2001] são exemplos de middlewares que utilizam esse tipo de representação. O primeiro tem um foco maior na introdução da ciência de contexto em aplicações pré-existentes, através da atuação de *widgets* ou serviços de interpretação do contexto. O segundo trabalho introduz o conceito de reflexão, que é a capacidade de um sistema de mudar seu comportamento ou se adaptar em função de determinada situação. Baseado nesse conceito, os autores criam um middleware reflexivo para aplicações móveis. O último desses trabalhos utiliza uma linguagem de marcação para implementar uma espécie de diretório de serviços para dispositivos móveis. Ryan [1999] apresenta um exemplo desse tipo de linguagem de marcação, criada especificamente para a troca de informações de contexto, chamada ConteXtML;

- Modelos Orientados a Objetos: utilizam os conceitos de hierarquia e reuso da orientação a objetos para tentar representar contextos com domínios complexos. Schmidt et al. [1999] apresenta um modelo orientado a objetos para extrair informações de contexto a partir de sensores. Bouzy & Cazenave [1997] utilizam

essa abordagem para definir quatro tipos de contexto: global, temporal, espacial e de finalidade.

- Modelos Lógicos: informações de contexto registradas através de predicados lógicos ou axiomas, que são normalmente utilizados em inteligência artificial para inferência. Abordagem proposta inicialmente em McCarthy [1993] com o objetivo de municiar entidades de inteligência artificial para racionalizar sobre o contexto e alcançar conclusões que transcendam o contexto registrado. Gray & Salber [2001] utilizam os modelos lógicos com foco na sensibilidade do contexto a partir de sensores para capturar propriedades e fenômenos do "mundo real", e Ghidini & Giunchiglia [2001] apresentam duas formas de extrair informações a partir do contexto: inferência por pontos de vista – pela combinação dos pontos de vista de vários usuários; e inferência por crenças – baseada no conhecimento individual.

- Ontologias: usam a inferência, assim como os modelos lógicos, mas a partir de fatos, conceitos e objetos e não apenas a partir de predicados lógicos. Estratégias utilizando esse modelo são comumente associadas à busca por conteúdo em redes P2P [Strang & Linnhoff-Popien, 2003; Chen, 2005]. Strang et al. [2003] demonstram uma linguagem de representação de contextos através de ontologias.

Embora existam divergências a respeito da possibilidade de transformação de ontologias em modelos lógicos e vice-versa [Horrocks et al., 2005], é importante citar a ferramenta Thea [Vassiliadis et al., 2009], que possibilita a tradução da linguagem de descrição de ontologias OWL2 (Web Ontology Language versão 2) para modelos lógicos interpretáveis usando a linguagem de programação funcional Prolog [Scowen, 1995]. Essa abordagem, que utiliza a programação funcional para representar modelos lógicos, é a base para o modelo adotado neste trabalho, por razões descritas no capítulo de solução (seção 4.2), dentre elas o poder de representação do modelo.

Para cada problema existe uma representação do contexto mais recomendada, variando em função da natureza do problema e, principalmente, do conhecimento que deve ser extraído a partir do contexto. Ele pode agregar diversos tipos de informação. Podem ser observações pontuais acerca do ambiente de computação atual, como bateria disponível em seu aparelho celular, velocidade de conexão com a Internet ou informação obtida a partir de seus sensores, como velocidade, temperatura ou frequência cardíaca do usuário. Dependendo da aplicação, o contexto pode incluir informações históricas, como o conteúdo cultural de preferência do usuário e o caminho utilizado diariamente para ir ao trabalho, ou ainda características pessoais como preferências culinárias, musicais e literárias.

A inferência de contexto é o processo de obtenção de conclusões mais úteis para a aplicação a partir de informações fornecidas pelos dispositivos e mensuradas através de seus sensores. Exemplos de inferência de contexto vão desde a localização semântica [Pradhan, 2000], informação obtida com base na posição geográfica do dispositivo porém com um valor semântico, como por exemplo "casa do usuário" ou "local de trabalho", até a detecção de atividades [Tapia et al., 2004; Kern et al., 2003], que significa determinar, a partir de alguns indícios, se o usuário está dormindo, trabalhando ou se exercitando, por exemplo. No caso das aplicações sociais, o contexto pode representar informações acerca das interações sociais passadas e presentes do usuário e, principalmente, informações que permitam inferir interações sociais futuras que o usuário possa protagonizar. A possibilidade de antecipar possíveis relacionamentos entre usuários de uma rede social, em função da similaridade dos seus contextos, é um pressuposto da solução adotada neste trabalho, que será melhor explicado na próxima seção (2.2).

2.2 Redes sociais

O conceito de rede social é utilizado para designar uma rede que representa os relacionamentos sociais de seus participantes. Embora existam divergências a respeito [Firth, 2004; Costa, 2012], considera-se que redes sociais podem ser utilizadas para representar relacionamentos inclusive entre pessoas que não se conhecem necessariamente. Uma rede social de pessoas que compartilhem interesses literários, que trabalhem em uma mesma empresa, ou que estejam assistindo a um mesmo filme na televisão, são exemplos de redes sociais nas quais o relacionamento não pressupõe o conhecimento pessoal entre as pessoas. Alguns relacionamentos são duradouros e fortes como os relacionamentos entre membros de uma família ou de amizade, outros somem com o tempo. As redes sociais são construídas para estudar fenômenos sociais [Kochen, 1989; Liljeros et al., 2001], compreender fluxos de informação [Kwak et al., 2010; Cheng et al., 2008], estudar o seu mecanismo de formação [Erdös & Rényi, 1959; Newman, 2001; Barabäsi et al., 2002] ou simplesmente advém de alguma aplicação que explora os relacionamentos que elas representam, como é o caso dos *sites* de redes sociais [Kirkpatrick, 2011; Hempel & Lehman, 2005]. A natureza humana influencia a formação de relacionamentos com diferentes objetivos nas redes sociais. Por exemplo, a possibilidade de criar contatos profissionais, manter contatos existentes com antigos colegas de universidade ou associar-se àqueles por quem se tem admiração. Esse aspecto psicológico na formação de relacionamentos proporciona a formação de redes com características particulares, dentre elas a tendência à formação de grupos, chamados de clusters ou

comunidades.

Newman [2001] ressalta o alto coeficiente de clusterização das redes sociais em comparação com o esperado para uma rede aleatória. Com base nesse indício, Barabäsi et al. [2002] comprovam através de uma abordagem baseada na evolução de redes sociais que a distribuição dos graus dos seus nós segue uma lei de potências. Na prática isso implica que nós com maior grau (número de conexões) têm maior probabilidade que novos nós se conectem a eles ao aderirem àquela mesma rede. Além disso, Leskovec et al. [2007] demonstram que, à medida que crescem as redes, a distância média, que é o número médio de arestas que separam os nós da rede, diminui – isso significa que, no caso da comunicação em redes sociais P2P descentralizadas, para redes maiores a comunicação entre os nós participantes tende a ser mais rápida, devido ao menor número de arestas que os pacotes devem percorrer. Essas características advém do mecanismo de formação dessas redes que segue uma lógica chamada de *preferential attachment*. Segundo esse conceito, quando novos nós aderem à rede existem preferências pessoais e critérios globais segundo os quais eles escolhem a quem irão se conectar, para maximizar a utilidade de seus relacionamentos.

Muitos estudos empíricos mostram que um critério que influencia na formação de novos relacionamentos é o número de conexões que um nó possui. Nesse caso, manifesta-se a característica percebida em [Barabäsi et al., 2002] de que os nós de maior grau recebem mais conexões novas e, consequentemente, formam clusters de usuários. Não é possível determinar todos os outros critérios pelos quais esses clusters se formam, embora existam alguns modelos que descrevem a evolução das redes sociais [Leskovec et al., 2008]. Entretanto, no caso das aplicações sociais móveis, o contexto dos usuários reflete seus hábitos e preferências e dois usuários com preferências e hábitos similares têm maior probabilidade de formarem relacionamentos em uma rede social. O conjunto de características que influenciam no comportamento social do usuário em termos da criação de novos relacionamentos é chamado de contexto social. Dentre as características que motivam a formação de relacionamentos, é possível citar a similaridade de interesses [Backstrom et al., 2006; Raacke & Bonds-Raacke, 2008; Boyd & Heer, 2006], experiência profissional [Davis et al., 2003], instituição de ensino frequentada e, principalmente, a proximidade física [Ghinita et al., 2007; Katz, 1994; Kraut et al., 1988] entre dois usuários.

Com base nas características que motivam a formação de relacionamentos em uma rede social, é possível construir uma infraestrutura para comunicação através de uma aplicação social móvel, que antecipe a formação de relacionamentos, agrupando usuários com maior probabilidade de se comunicarem. A solução proposta utiliza a similaridade entre o contexto dos usuários para antecipar os relacionamentos entre eles

e constrói uma camada *overlay* de agentes autônomos para suportar a comunicação através de aplicações sociais móveis.

2.3 Agentes autônomos

A implementação da camada *overlay* proposta neste trabalho trata cada usuário como um agente autônomo. Segundo Franklin & Graesser [1997], um agente autônomo é um sistema imerso em um ambiente que o percebe e age sobre ele, ao longo do tempo, em busca de seus próprios objetivos e para afetar suas futuras percepções do ambiente. Existem quatro características que justificam a escolha dos agentes autônomos para a arquitetura proposta neste trabalho:

- autonomia: operam sem a intervenção direta humana e possuem controle sobre seu próprio estado e ações;
- sociabilidade: interagem com outros agentes e, possivelmente, com o usuário, através de alguma linguagem de comunicação;
- reatividade: têm percepção das alterações no ambiente, através da coleta de informações, contexto e interação de usuários, interação com outros agentes, e respondem a essas mudanças;
- proatividade: não reagem simplesmente ao ambiente, pois tomam iniciativas em função de determinado objetivo.

Para que a arquitetura proposta seja escalável e permita a construção e o bom funcionamento de redes com números suficientemente grandes de participantes é desejável que os dispositivos sejam autônomos, pois a dependência de uma entidade de coordenação pode, possivelmente, limitar a extensão de sua rede. A independência de intervenção humana proporcionada pela autonomia é também importante por se tratarem de dispositivos móveis, e embora permaneçam ligados durante grande parte do dia, a interação do usuário é mais esporádica. A reatividade e a proatividade são características essenciais para a manutenção da rede em função de falha ou desligamento de participantes e para a reorganização dos participantes em função de mudanças de contexto. Esses agentes autônomos, ao se organizarem em diferentes topologias para beneficiar a aplicação social, formam uma camada *overlay peer-to-peer* e representam bem o comportamento humano através da constituição de relacionamentos e pela interação com outros usuários.

2.4 Camada *overlay peer-to-peer*

Existem abordagens para redes sociais móveis, que modelam aspectos como a conexão oportunista e a mobilidade. Entretanto, com a difusão dos padrões de conexão de dados através de redes sem fio e principalmente das redes celulares, é possível implementar uma rede social móvel usando uma camada *overlay peer-to-peer* sobre a Internet. Uma camada *overlay* p2p é uma abstração lógica sobreposta à infraestrutura de conexão física entre os nós participantes, que permite modelar a comunicação utilizando topologias independentes das conexões físicas entre os participantes. Essa flexibilidade na topologia *overlay* e a descentralização inerente às redes p2p permitem a escalabilidade necessária para a arquitetura, suportam a constante conexão e desconexão de nós, a busca distribuída de conteúdo e, principalmente, a cooperação autônoma entre os agentes. Dados os grafos $I(V, A)$ e $O(V', A')$, é possível caracterizar a camada *overlay* P2P da seguinte forma:

$$(V' \subseteq V) \tag{2.1}$$

$$(v_i, v_j) \in V', \exists path(v_i, \ldots, v_j) \in A \implies (v_i, v_j) \in A' \tag{2.2}$$

Nessa representação I é o grafo da Internet e O a camada *overlay*. A primeira condição advém do próprio conceito de *overlay*, pois os nós de uma camada sobreposta a uma rede são um subconjunto dos nós da rede original. Em outras palavras, todos os nós que compõem a camada *overlay* estão conectados à Internet. A segunda condição diz que: se existe um caminho entre um par de nós $(v_i, v_j) \in V'$ no grafo da Internet, então esse caminho será representado através de uma aresta entre esses dois vértices no grafo da camada *overlay*. Se fizermos a suposição de que todos os vértices do grafo da Internet são alcançáveis em um número finito de saltos, então o grafo da camada *overlay* será um grafo completo. É possível fazer essa suposição por que todo dispositivo com acesso à Internet possui necessariamente um endereço IP válido, mesmo que seja um fornecido dinamicamente pela operadora de celular e, assume-se que todo dispositivo com endereço IP válido é alcançável a partir de qualquer outro. No entanto, para utilizar essa hipótese o protocolo P2P deve lidar com a questão da mudança de endereço IP de um nó.

2.5 Clusterização

A justificativa para o uso de uma camada *overlay peer-to-peer* é a sua flexibilidade para adquirir diferentes topologias. Para o caso específico deste trabalho, a topologia desejada é representada por um conjunto de clusters, cujos integrantes apresentam uma razoável similaridade de contextos. A clusterização é uma estratégia muito utilizada na classificação de documentos [Steinbach et al., 2000; Zamir & Etzioni, 1998], nesse caso cada cluster possui um centróide, calculado a partir da distribuição de termos dos documentos que o integram. O conceito de centróide vem da geometria, na qual ele representa o centro de massa de um objeto. Analogamente, nos algoritmos de clusterização, o centróide é representado através da distribuição de probabilidades de elementos que compõem um grupo. Esses elementos podem ser as palavras de um conjunto de documentos, no caso dos algoritmos de classificação, bases nitrogenadas de DNAs ou ainda dados de sensores em uma rede. Através do uso dos centróides, os algoritmos de clusterização decidem em qual cluster serão inseridos novos documentos, usuários ou dados, com o intuito de minimizar a diferença entre os integrantes dos clusters.

É possível modelar a clusterização de usuários semelhantes através do problema do particionamento de grafos [Garey et al., 1976]. O problema do particionamento de grafos é antigo com origens na física e pode ser usado na atribuição de circuitos eletrônicos a placas de circuitos, com a intenção de minimizar as conexões entre as placas, na clusterização de dados, para a paralelização do processamento e no particionamento de grafos da Web, para identificar tópicos e usá-los nas buscas por documentos. O problema da clusterização de usuários usando a similaridade de contextos também pode ser descrito usando o problema do particionamento de grafos.

Dado um grafo ponderado $G = \{V, E\}$, e um conjunto de clusters K, a variável x_v^k é definida da seguinte maneira:

$$x_v^k = \begin{cases} 1 & \text{sse } v \in \text{cluster } k \\ 0 & \text{caso contrario} \end{cases} \tag{2.3}$$

$$\sum_{k=1}^{k} x_v^k = 1, \forall v \in V. \tag{2.4}$$

Utilizando a variável de modelagem $y_{(u,v)}$, tal que:

$$y_{(u,v)} = \begin{cases} 1 & \text{sse } u \in \text{cluster } k \text{ e } v \notin \text{cluster } k \\ 0 & \text{caso contrario.} \end{cases} \tag{2.5}$$

O objetivo é:

$$\min \sum_{\forall (u,v) \in E} w_{(u,v)} y_{(u,v)} \tag{2.6}$$

segundo as restrições:

$$x_u^k - x_v^k \leq y_{(u,v)}, \quad \forall (u,v) \in E, \forall k \in K \tag{2.7}$$

onde $w_{(u,v)}$ é o peso das arestas.

A variável binária x_v^k, instanciada segundo a fórmula 2.3 registra a presença do nó v no cluster k, assumindo os valores 1 ou 0, para representar a participação ou não do nó. A restrição 2.4 garante que um nó deverá pertencer a um e apenas um cluster. A variável $y_{(u,v)}$ assume o valor 1 para todas as arestas que atravessam de um cluster a outro, segundo a fórmula 2.5 e permite calcular o custo do particionamento, através da soma dos pesos dessas arestas que transpõem os clusters. Esse custo está representado na função 2.6, que determina o objetivo de minimizar o custo do particionamento, segundo a restrição 2.7, que relaciona a variável $y_{(u,v)}$, com a variável x_v^k.

Se for calculada a distância semântica de contextos entre cada par de usuários, e registrá-la como uma matriz de pesos das arestas entre eles, é possível determinar o melhor particionamento de um grafo, que maximize a semelhança entre usuários de um mesmo cluster. No entanto, o problema do particionamento de grafos é NP-completo [Ding et al., 2001], assim sua solução ótima não é conhecida em tempo polinomial. Algumas das heurísticas utilizadas para a solução desse problema são baseadas no algoritmo Ford-Fulkerson, derivado do teorema do corte mínimo e fluxo máximo [Ford Jr & Fulkerson, 1958], outras utilizam algoritmos gulosos para obter melhores particionamentos a partir de uma configuração inicial [Kernighan & Lin, 1970] e finalmente existem heurísticas que se baseiam na clusterização pelo cálculo de centróides [Ng et al., 2002]. Dentre essas abordagens, aquela que utiliza a clusterização é a mais apropriada para a implementação distribuída, pois independe de uma visão global do grafo, para o cálculo do custo do particionamento ou a obtenção gulosa de uma nova solução. A seção 4.1 explica a abordagem de clusterização ciente de contexto proposta neste trabalho.

Capítulo 3

Trabalhos Relacionados

Aplicações sociais móveis apresentam requisitos da computação ubíqua, das redes sociais e dos algoritmos distribuídos. Este capítulo descreve a evolução dessas três áreas, apresenta os trabalhos com objetivos semelhantes e explica as limitações e qualidades dos mesmos que podem ser aproveitadas por uma arquitetura para desenvolvimento de aplicações sociais móveis.

3.1 Evolução da pesquisa na área

A solução proposta neste trabalho utiliza algoritmos distribuídos para garantir requisitos de computação ubíqua a aplicações em redes sociais. A confluência entre as três áreas de pesquisa – algoritmos distribuídos, computação ubíqua e redes sociais – é natural se observarmos a evolução desses tópicos ao longo do tempo.

Os algoritmos distribuídos tiveram sua origem nos primeiros protocolos de comunicação entre computadores [Postel, 1980; Droms, 1993], sendo amplamente aplicados também na comunicação entre processadores [Pancake & Utter, 1991], sensores [Guo et al., 2001; Shah & Rabaey, 2002] e dispositivos móveis [Perkins, 1998; Solomon, 1997], na comunicação sem fio [Garcia-Luna-Aceves & Madruga, 1999] e através das redes celulares [Valkó, 1999]. Um tema que concentra muitos esforços de pesquisa, pela sua ampla aplicabilidade, são as redes *peer-to-peer*, tanto para representar a conectividade física entre dois dispositivos quanto para modelar aspectos de nível mais alto utilizando camadas *overlay peer-to-peer* sobrepostas à alguma infraestrutura física de conexão. Essas camadas *overlay* evoluíram de arquiteturas não-estruturadas, nas quais as buscas por conteúdo são realizadas através de protocolos de alagamento (*flooding*) [Adar & Huberman, 2000], para arquiteturas estruturadas, utilizando algoritmos distri-

buídos de auto-organização e difusão de chaves, que proporcionam buscas por conteúdo mais eficientes [Stoica et al., 2001].

O termo computação ubíqua é definido por Weiser [1993] como uma forma de incrementar o uso dos computadores através da disposição dos mesmos no ambiente físico, porém tornando-os invisíveis para o usuário. Essa definição tem origem nas pesquisas com *smart spaces*, ambientes controlados nos quais se tentava interagir com o usuário da forma mais imperceptível e menos onerosa possível [Schilit et al., 1994; Kidd et al., 1999]. Entretanto, com a disseminação do uso dos dispositivos de computação móvel, principalmente os *smartphones*, a computação ubíqua focou-se no uso desses dispositivos de forma mais transparente e com menor dispêndio de atenção do usuário [Chen & Kotz, 2000]. Como alternativa para adaptar a computação às características do ambiente, estado e preferências do usuário, intensificou-se o uso de sensores, através da computação vestível (*wearable computing*) [Kern & Schiele, 2003; Krause et al., 2006] e instalados nos próprios *smartphones*, e.g. GPS, bússola e acelerômetro. A adaptação das aplicações a essas características citadas é chamada de ciência de contexto e é a principal forma utilizada para obter aplicações ubíquas, capazes de se mesclar no cotidiano do usuário.

As redes sociais gradualmente deixaram de ser conceituais, utilizadas inicialmente para o estudo do comportamento humano [Christakis & Fowler, 2007] e dos aspectos psicológicos por trás da formação de relacionamentos entre as pessoas [Newman, 2001; Barabäsi et al., 2002], e se tornaram plataformas para aplicações sociais onipresentes [Raacke & Bonds-Raacke, 2008]. Costa [2012] descreve a evolução das redes sociais através de três gerações diferentes. Na primeira delas, as redes sociais eram baseadas na comunicação pessoal e nos aplicativos de mensagens instantâneas (*instant messengers*). A segunda geração surgiu a partir do intuito de se replicar as redes de afinidades e de conhecidos das pessoas, ou seja, elas tinham como objetivo representar redes sociais "reais" em ambientes virtuais. Exemplos dessas redes são o *Facebook*, *Orkut* e o *Friendster*. Mais tarde algumas dessas redes evoluíram para uma terceira geração, de sistemas de criação e aquisição de experiências e conteúdo. Algumas outras já surgiram com esse intuito, como é o caso do *LinkedIn* e do *MySpace*, compondo também essa terceira geração de redes sociais.

A confluência dessas três áreas foi influenciada pelos dispositivos móveis e sua onipresença os credencia como principais plataformas de acesso às redes sociais.

3.2 Camadas *overlay peer-to-peer*

Por permitirem a criação de uma abstração lógica e dinâmica sobre uma infraestrutura de conexão física, como é o caso dos *links* de Internet, as camadas *overlay* são muito aplicadas aos problemas de busca de conteúdo e algoritmos distribuídos em geral. A busca por conteúdo é também um fator chave da arquitetura proposta neste trabalho e o conteúdo avaliado por essas buscas é o contexto dos usuários.

Os protocolos P2P mais conhecidos e utilizados na busca de conteúdo podem ser divididos em duas gerações. A primeira delas é composta pelo protocolo Gnutella [Ripeanu, 2001] – que se difundiu como o precursor dos protocolos P2P de compartilhamento de arquivos, embora sua escalabilidade fosse limitada em função do uso de *broadcasting* na busca do conteúdo – e o Freenet [Clarke et al., 2001] – com foco na publicação, replicação e obtenção de informações de forma distribuída e anônima. A segunda geração é constituída por protocolos inspirados na eliminação das limitações dos protocolos da primeira geração e, principalmente, cujas buscas possuem um limite superior para o número de saltos necessários, proporcionado pelo uso de camadas *overlay* estruturadas. São exemplos desses protocolos o Chord [Stoica et al., 2001], o Pastry [Rowstron & Druschel, 2001], o Tapestry [Zhao et al., 2001] e o CAN [Ratnasamy et al., 2001].

Embora esses protocolos apresentem um bom desempenho nas buscas distribuídas e boa tolerância a falhas, eles não são aplicados aos problemas cujas buscas devem retornar resultados aproximados e seus mecanismos de auto-organização não aproximam os *peers* que armazenam conteúdo semelhante. Adicionalmente, a publicação de informação nesses protocolos não é preparada para armazenar um conteúdo tão dinâmico como é o contexto de cada um desses *peers*. Uma tentativa de estender o protocolo Gnutella para armazenamento de informações de contexto foi aplicada em Gu et al. [2005]. Nesse trabalho, os autores propõem uma arquitetura com múltiplas camadas *overlay* representando tipos diferentes de contexto. Além disso, cada camada é composta por clusters semânticos de nós participantes e um algoritmo de roteamento de buscas foi proposto para permitir o uso dessas abstrações. Embora a solução proposta nesse trabalho seja interessante para o armazenamento do contexto e a clusterização proposta aproxime os nós cujo contexto é semelhante, ela não se aplica às redes sociais e não prevê a utilização de contextos sociais para influenciar nessa clusterização.

Este trabalho apresenta uma camada *overlay peer-to-peer* que possibilita a execução de buscas aproximadas através da clusterização de usuários com contextos semelhantes. Essa forma de estruturação da rede é semelhante ao problema do particionamento de grafos, detalhado a seguir.

3.3 Particionamento de Grafos

O problema do particionamento de grafos é encontrado em muitas áreas de pesquisa, dentre elas a clusterização de documentos [Dhillon, 2001] e dados [Ding et al., 2001] – que aproximam aqueles semelhantes na forma de um cluster, assim como é o objetivo da arquitetura proposta neste trabalho – as redes de sensores [Ghiasi et al., 2002] – onde a correlação dos dados é proporcionada pela atuação dos líderes dos clusters através de algoritmos distribuídos e protocolos de comunicação – e as redes sociais [Newman, 2006] – principalmente para o estudo das estruturas de comunidade formadas nessas redes.

Por se tratar de um problema NP-completo, existem também heurísticas propostas para esse problema, dentre elas uma solução baseada no teorema do corte mínimo/fluxo máximo, apresentada em Ding et al. [2001], uma heurística gulosa apresentada em Battiti & Bertossi [1999], um algoritmo genético de [Bui & Moon, 1996] e até uma biblioteca de software com diferentes heurísticas para o problema [Preis & Diekmann, 1997]. Essas heurísticas são aplicáveis, principalmente, para os casos em que os grafos particionados são estáveis, e as distâncias entre os nós são constantes e facilmente calculáveis. Entretanto, para particionar grafos em função da diferença entre os contextos dos usuários, representados pelos nós dos grafos, é mais interessante utilizar algoritmos de clusterização. A razão para tal é a capacidade desses algoritmos de modificar rapidamente o particionamento encontrado, caso algum dos nós sofra alguma alteração em seu contexto, fato muito comum quando se lida com dispositivos móveis.

3.4 Aplicações Sociais Móveis

Além deste trabalho, existem outros que lidam com a confluência das áreas de algoritmos distribuídos, computação ubíqua e redes sociais e tentam estabelecer metodologias para o desenvolvimento de aplicações sociais móveis.

Pietiläinen et al. [2009] apresentam um middleware para o desenvolvimento de aplicações sociais móveis que se comuniquem oportunisticamente através de redes Bluetooth. González et al. [2006] apresentam um sistema baseado em colisão de partículas para construir redes sociais de agentes móveis. Embora o comportamento das partículas seja diferente dos padrões de mobilidade humana pela grande aleatoriedade na sua movimentação, os autores demonstram obter redes com características semelhantes àquelas conhecidas para redes sociais móveis. Bottazzi et al. [2006] diferenciam os usuários de seu middleware em usuários comuns ou *ego users*, que são usuários

que podem criar redes sociais a partir de especificações semânticas para contextos em comum. A abordagem dos autores difere deste trabalho pelo fato da formação dos grupos acontecer sob demanda. Entretanto, essa formação de grupos também é ciente de contexto e representa uma boa referência prática para a proposta de clusterização, descrita na seção 4.1. Gupta et al. [2009] apresentam um middleware para a captura de informações de estado de usuários em aplicações sociais móveis, além da correlação desses dados para detectar padrões geo-sociais de comportamento coletivo, um conceito introduzido pelos autores que pode ser substituído pelo conceito de contexto coletivo, utilizado neste trabalho. Mokhtar et al. [2009] definem três níveis de descentralização para a implantação de uma rede social móvel: totalmente centralizada, semi-distribuída – através de uma camada *overlay peer-to-peer* e totalmente distribuída. Em seguida, eles implementam uma arquitetura semi-distribuída, na qual são eleitos líderes que agem sobre os outros participantes através da recomendação de possíveis conexões com outros usuários, das quais eles possam se beneficiar.

3.5 Análise

Uma arquitetura para o desenvolvimento de aplicações sociais móveis poderia se beneficiar dos resultados dos trabalhos descritos acima. Ela deve ser descentralizada, assim como a solução apresentada por Pietiläinen et al. [2009], para permitir a escalabilidade e a comunicação entre um grande número de participantes sem onerar entidades centralizadoras. No entanto, ela não pode se restringir apenas aos modelos de redes sociais formadas oportunisticamente, como tratadas no modelo desses autores, uma vez que a maioria das redes sociais móveis apresentam relacionamentos de longa duração entre seus usuários.

A organização dos participantes deve ser semelhante à rede social que a arquitetura suporta, assim como no trabalho de González et al. [2006]. Dessa forma, a arquitetura irá beneficiar a comunicação entre usuários relacionados socialmente ou com maior probabilidade de se relacionarem, que ocorrerá de forma mais rápida pela sua proximidade em termos de arestas percorridas na camada *overlay*. A arquitetura proposta por Mokhtar et al. [2009] tenta antecipar futuros relacionamentos na rede, embora utilize apenas a proximidade física e a quantidade de relacionamentos em comum entre os participantes para antever esses relacionamentos.

A formação de grupos deve tratar a semelhança semântica entre os contextos dos usuários, assim como no trabalho de Bottazzi et al. [2006]. Apesar dos autores tratarem a formação de grupos a partir iniciativa de um usuário, se realizada de forma automá-

tica, a clusterização permite a execução de buscas aproximadas, capazes de retornar um número determinado de usuários com o contexto semelhante àquele desejado pela aplicação.

A arquitetura deve tratar questões relativas ao contexto, permitir a correlação dos dados e apresentar uma interface com as aplicações através de uma API (*Application Programming Interface*) ou interface de programação para aplicações, pois dessa forma pode contribuir para o desenvolvimento de aplicações sociais móveis. O protótipo implementado em Gupta et al. [2009] engloba essas características, embora centralize as funcionalidades de descoberta de usuários, formação de grupos e armazenamento do contexto em entidades separadas dos usuários, constituindo uma arquitetura pouco escalável, mesmo com o uso de memória *cache*, para a resposta mais ágil às chamadas de procedimento no servidor.

A arquitetura proposta neste trabalho aborda os aspectos citados acima, embora apresente as suas próprias limitações, apresentadas no capítulo 6. Existem ainda vários outros desafios ao desenvolvimento dessas aplicações. Toninelli et al. [2010] e Buchegger & Datta [2009] citam alguns deles. Entretanto, acredita-se que a solução proposta neste trabalho beneficiará o desenvolvimento de aplicações sociais móveis, principalmente proporcionando mecanismos de buscas aproximadas em uma camada *overlay* ciente de contexto e com viés social.

Capítulo 4

CAWTEC

Este capítulo apresenta a CAWTEC uma arquitetura ciente de contexto para o desenvolvimento de aplicações sociais móveis.

4.1 Arquitetura

Para resolver o problema da construção de redes *peer-to-peer* para o suporte a aplicações sociais móveis propõe-se uma arquitetura com os seguintes componentes (figura 4.1):

- um mecanismo de inferência de contexto – representado como um componente modular, pois foi implementado de forma acoplável à arquitetura, permitindo o uso de outros motores de inferência disponíveis;
- a camada *overlay peer-to-peer* – organiza os usuários em função da similaridade de seus contextos;
- o protocolo P2P – que possibilita a criação do *overlay*, implementando os mecanismos de comunicação, busca, adesão, clusterização e monitoramento de contextos;
- a interface de desenvolvimento – uma interface programática entre as aplicações e a arquitetura.

Não fazem parte do escopo deste trabalho, embora estejam também representadas na figura, as aplicações, que utilizarão a interface de desenvolvimento para explorar os relacionamentos entre os usuários da rede social, correlacionar dados através de contextos coletivos e atuar sobre os usuários e o ambiente do qual fazem parte. Cada um desses componentes será explicado em detalhe nas próximas seções deste documento. A seção 4.6 relaciona todos os componentes através da descrição das aplicações sociais

móveis que podem se beneficiar desta arquitetura, a metologia para o seu desenvolvimento e a aplicação de exemplo, chamada *Social Campi*.

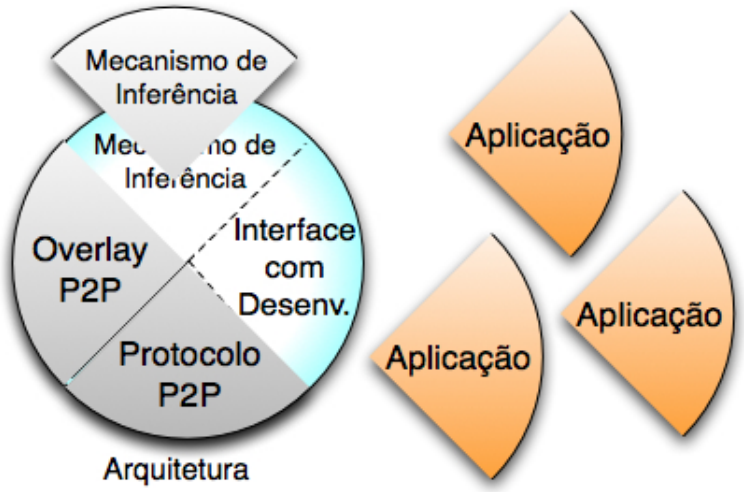


Figura 4.1. Arquitetura

4.2 Mecanismo de inferência de contexto

Antes de passar pelo processo de inferência, o contexto de um usuário é representado por um conjunto de valores e métricas mensuráveis e fornecidas pelo seu dispositivo móvel ou outro sensor utilizado em determinada aplicação. A inferência de contexto corresponde a submeter esses dados à interpretação usando o conhecimento do domínio, alimentado e armazenado no motor de inferência.

Embora seja recomendado o uso de linguagens de descrição de conhecimento a partir de ontologias compartilhadas, por permitirem o reúso das entidades e relacionamentos, optou-se por utilizar uma abordagem baseada em uma linguagem funcional, o Prolog [Scowen, 1995].

O processo de descrição do domínio através de predicados lógicos em Prolog é geralmente mais difícil do que descrevê-lo através de ontologias. No entanto, pela ampla difusão da linguagem Prolog e de suas implementações e pela sua padronização, através da norma ISO/IEC 13211-1 [Scowen, 1995], motores de inferência implementados em Prolog são mais confiáveis do que aqueles que utilizam ontologias. Além disso, embora existam divergências à respeito da possibilidade de se traduzir ontologias para modelos

Tabela 4.1. Exemplos de relações da OWL2 representadas em Prolog

OWL	Prolog
<owl:Class rdf:about='Child'> <rdfs:subClassOf rdf:resource='Person'/> </owl:Class>	child(X) :- person(X).
<owl:ObjectProperty rdf:about='hasHusband'> <rdfs:subPropertyOf rdf:resource='married'/> </owl:ObjectProperty>	hasHusband(X,Y) :- married(X,Y).
<rdf:Description rdf:about='hasGrandparent'> <owl:propertyChainAxiom rdf:parseType='Collection'> <owl:ObjectProperty rdf:about='hasParent'/> <owl:ObjectProperty rdf:about='hasParent'/> </owl:propertyChainAxiom> </rdf:Description>	hasGrandparent(X,Z) :- hasParent(X,Y), hasParent(Y,Z).

lógicos [Horrocks et al., 2005], Prolog é uma linguagem Turing completa, enquanto as linguagens de ontologias, dentre elas a OWL e a OWL2, não são [Berners-Lee & Mendelsohn, 2006]. Isso significa que o poder computacional da linguagem Prolog é maior, uma vez que ela é capaz de simular qualquer máquina de Turing com uma fita, e por princípio, qualquer computador determinístico sem paralelismo [Tärnlund, 1977]. No entanto, a maior razão pela qual optou-se por utilizar modelos lógicos para a representação do contexto e sua inferência é a existência de interpretadores lógicos eficientes e disponíveis para o uso imediato. A implementação de Prolog utilizada neste trabalho foi o SWI Prolog [Wielemaker, 2003], por prover métodos para a implementação de um servidor web, através do qual a comunicação com a arquitetura possa ocorrer de uma forma modular e desacoplada. A abordagem usando linguagens funcionais é apropriada para representar o conhecimento necessário à diversas aplicações, dentre elas a aplicação *Social Campi*, implementada neste trabalho para demonstrar o funcionamento do protocolo e da camada *overlay peer-to-peer*. Além disso, caso o desenvolvedor de uma aplicação deseje utilizar alguma ontologia pública, ele poderá traduzí-la, usando uma representação em Prolog, se for utilizar o motor de inferência implementado nesta arquitetura, ou acoplar seu próprio mecanismo de inferência através também de um *webservice*. A tabela 4.1 mostra exemplos de possíveis traduções de algumas regras da ontologia OWL2, encontradas em Hitzler et al. [2009], para a linguagem Prolog.

A opção pelo modelo lógico para representação dos contextos também é justificada pela facilidade em representar problemas simples, como o cenário de uso deste trabalho, e pela maior facilidade em criar, aleatoriamente, modelos para descrever pro-

blemas hipotéticos, necessários para os experimentos de simulação. O algoritmo para criação desses modelos hipotéticos e os experimentos que os utilizam estão descritos no capítulo 5.

4.3 Camada *overlay peer-to-peer* ciente de contexto

A partir da similaridade entre os contexto dos usuários, constrói-se os clusters, ou grupos, da camada *overlay peer-to-peer*. Cada um desses clusters possui um líder, que armazena a informação do centróide de seu grupo. Para calcular o centróide do grupo, o líder avalia o contexto de cada um dos participantes segundo os predicados possíveis para o domínio do problema. Isso significa que o líder irá avaliar, por exemplo, onde aquele usuário mora, qual o seu local de trabalho ou onde ele está no momento e a atividade que está desempenhando. A avaliação desses predicados é registrada no centróide através de uma distribuição de probabilidades. Se em um mesmo grupo, todos os usuários trabalharem em um mesmo local, ou morarem em uma mesma cidade, então esses predicados estarão representados no centróide com a probabilidade de 100%. No entanto, à medida que forem adicionados a esse cluster usuários que morem em outras cidades, essa probabilidade deixará de ser de 100%.

Para construir a camada *overlay* ciente de contexto, criou-se um algoritmo de clusterização. Cada nó, ao aderir à camada *overlay*, avalia qual cluster apresenta a maior probabilidade de que existam usuários com o contexto semelhante ao seu, caso essa probabilidade seja satisfatória, segundo um índice de tolerância t, ele adere àquele cluster, caso contrário, ele se torna o líder de um novo cluster. Segundo esse processo de decisão, o índice de tolerância t tem influência direta no número de clusters formados na rede. No pior caso, quando t=0, um nó aderirá a um cluster apenas se todos os usuários ali possuírem os contextos idênticos ao seu. O capítulo 5 demonstra a variação do número de clusters em função do índice de tolerância t e do número de predicados do domínio p.

A eleição dos líderes dos clusters não faz parte do escopo desse trabalho, mas pode influenciar no desempenho da solução. É ideal que os líderes de cluster possuam característica de hardware menos limitantes, embora o capítulo 5 demonstre que, para o protótipo de aplicação Social Campi, o ônus de ser um líder de cluster não impossibilita que um dispositivo móvel, com as características semelhantes àquelas encontradas atualmente no mercado, possa desempenhar esse papel. Outro critério possível para a eleição do líder pode ser o seu número de relacionamentos na rede social, pois isso de-

termina maior probabilidade de que novos participantes se relacionem com ele. Dessa forma, a probabilidade de que novos participantes da camada *overlay* já possuam um relacionamento social pré-definido com algum dos líderes de cluster é maior, o que irá contribuir para que a camada *overlay* construída reflita melhor as interações na aplicação social que ela suporta.

4.4 Protocolo da camada *peer-to-peer* ciente de contexto

Para permitir a organização da camada *overlay* e as operações que ele proporciona para o desenvolvedor de uma aplicação, foi implementado um protocolo P2P ciente de contexto. Os participantes podem assumir nove estados diferentes segundo o protocolo, e as transições entre esses estados acontecem em função do recebimento de determinadas mensagens ou por vontade própria do participante. Esta seção apresenta a descrição de cada estado e suas transições, ambos representados na figura 4.2.

- Fora: este é o estado dos nós que não participam da camada *overlay*. A partir desse estado, por decisão própria de participar, o nó deverá consultar os centróides dos clusters e definir ser irá inaugurar um novo cluster ou aderir a algum já existente;

- Consultando líderes: é nesse estado que o nó decide se existe algum cluster ao qual ele deseja aderir, o que corresponde à transição para o estado "Aderindo", ou se ele criará um novo cluster, caso no qual ele fará a transição para o estado "Líder";

- Aderindo: ao determinar que deseja participar de um cluster, um nó envia uma mensagem de adesão ao seu líder, que deve autorizar a sua entrada. A sua adesão ao cluster, no entanto, provoca a transição do líder para o estado "Calculando Centróide", para que o mesmo passe a refletir a existência daquele novo participante;

- Ocioso: ao aderir a um cluster, o nó assume o estado ocioso em prontidão para auxiliar nos métodos de desenvolvimento de aplicações proporcionados pela arquitetura.

- Líder: Ao receber uma mensagem de busca por um determinado nó, o líder do cluster provoca um *flooding* entre os participantes de seu grupo, para determinar

aquele que mais se assemelha à busca que lhe foi requisitada. Nesse momento, ele passará temporariamente para o estado "Coletando contextos", até que receba a resposta de todos os participantes ou atinja uma temporização, caso no qual ele considerará que os nós que não responderam falharam. A outra transição possível para um nó líder é para o estado "Calculando centróide", que acontece periodicamente, para detecção de falhas nos participantes ou para o monitoramento de seus contextos, para informar alguma parte interessada, ou é provocada pela adesão de um novo nó ao cluster, gerando a necessidade de recalcular o centróide. A partir desse estado, um líder pode decidir, a qualquer momento, sair da camada *overlay*, caso no qual os nós participantes de seu cluster irão eleger um novo líder;

- Calculando centróide: o estado de cálculo do centróide e o estado de coleta dos contextos são semelhantes, pois é neste momento que o líder do cluster consulta o mecanismo de inferência. No estado de cálculo do centróide, ele o faz para determinar a distribuição de probabilidades de todos os predicados possíveis, que compõem o contexto dos usuários;

- Coletando contextos: assim como no estado de cálculo do centróide, aqui o líder consulta o mecanismo de inferência, porém, nesse caso, com o intuito de determinar qual participante é o melhor candidato para determinada busca em execução;

- Buscando nó: enquanto está ocioso como integrante de um grupo ou como líder, um nó pode decidir se comunicar com outro, nesse caso, ele assume este estado até que sua busca seja respondida;

- Saindo: ao decidir deixar, graciosamente (*gracefully*), um cluster, o nó envia uma mensagem correspondente ao líder e assume este estado até que receba a confirmação de que não pertence mais àquele grupo.

A partir da máquina de estados do protocolo e das possíveis transições, construiu-se um modelo probabilistico na linguagem Prism [Kwiatkowska et al., 2002], uma linguagem de *model-checking* que permite a avaliação de algumas propriedades. O modelo construído demonstra que todos os estados podem ser alcançados em até no máximo 5 transições, a partir do estado em que o nó ainda não aderiu à camada *overlay*. Embora a validação através de *model checking* não seja o objetivo principal deste trabalho, pois para tal foi implementado um simulador discreto de eventos, descrito no capítulo 5,

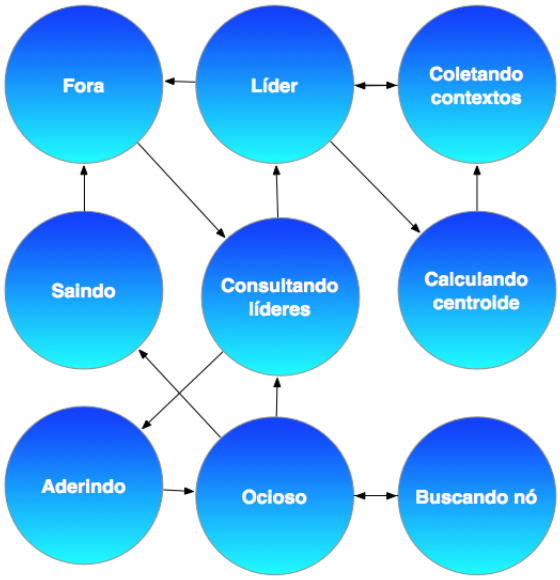


Figura 4.2. Máquina de estados do protocolo

essa abordagem nos permite verificar a seguinte propriedade do modelo, que não pode ser verificada através de simulação:

$$\forall(s_0, s_1, \ldots, s_i)|s_i \in S \quad \exists(s_i, \ldots, s_f)\forall s_f \in S \tag{4.1}$$

S é o conjunto de todos os estados possíveis para os nós e $(s_0, s_1, \ldots, s_i)$ é uma sequência qualquer de transições de estados. A propriedade diz que em qualquer ponto da execução do protocolo, existirá pelo menos uma sequência de transições que levará o nó a qualquer um dos estados possíveis. Na prática, isso significa dizer que não haverá uma sequência de transições para um nó que o levará para um estado ou conjunto de estados possíveis, dos quais não poderá retornar. Outras propriedades desejadas para o protocolo estão descritas no capítulo 5 e foram constatadas através da simulação.

A utilidade de um protocolo está atrelada às funcionalidades que ele proporciona à camada de software posicionada sobre o mesmo. Essa camada é chamada de API (Application Protocol Interface) ou interface programática para aplicações.

4.5 Interface com o Desenvolvimento de Aplicações (API)

Para facilitar o desenvolvimento de aplicações sociais móveis, é imprescindível que a arquitetura possua uma interface de conexão com as aplicações que permita o completo

Tabela 4.2. Implementação do *Delegate*

Protocolo RetornoChamadas

```
@protocol RetornoChamadas <NSObject>

- (void)retornoBusca: (NSMutableData*)mensagem;
- (void)retornoAdesao: (NSMutableData*)mensagem;
- (void)retornoMonitoramento: (NSMutableData*)mensagem;

@end
```

uso das funcionalidades da arquitetura. Antes de descrever os métodos da API é necessário introduzir o padrão de projeto de software chamado *Delegate*. Esse padrão de projeto pressupõe a existência de dois objetos distintos, um deles recebe a requisição e delega a operação para o outro, o seu *"delegate"* [Gamma et al., 1994]. Em Objective-C, o *delegate* implementa um protocolo e em Java ele implementa uma interface, que são conjuntos de assinaturas de métodos.

No caso da arquitetura implementada, criou-se um objeto que recebe o retorno das chamadas dos métodos da API, representados na tabela 4.3, e delega o tratamento dessas requisições através de uma interface chamada `RetornoChamadas`, representada na tabela 4.2. O desenvolvedor que deseja utilizar os métodos de busca, adesão e monitoramento do contexto, disponíveis através da API da arquitetura, deve criar o objeto *delegate* que implemente essa interface.

A tabela 4.3, apresenta as assinaturas dos métodos disponíveis para o uso pela aplicação. Para a adesão e para a busca é necessário fornecer o contexto do nó que deseja participar da camada *overlay* e daquele que se deseja encontrar nessa camada. O objeto contexto foi implementado como o conjunto de predicados lógicos válidos para aquele nó. Por exemplo, no caso de uma aplicação de monitoramento médico, o contexto de um usuário poderia ser composto pelos predicados "`idade(X) :- between(20, 25, X).`" e "`predisposição(X) :- diabetes, hipertensão.`", dentre outros vários predicados possíveis para esse domínio de problemas. Um exemplo melhor da representação do contexto está descrito na próxima seção, através da aplicação *Social Campi*.

O método `inscreve` recebe como parâmetro o endereço IP do nó cujo contexto deve ser monitorado. Caso esse endereço não seja conhecido, a aplicação deverá obtê-lo através do método de `busca`. Os parâmetros de tolerância, em ambos os casos, determinam o quão similar deverão ser os contextos. No caso da busca, a tolerância

Tabela 4.3. Assinaturas dos métodos da interface

Adesão	Saída

`(void) adere(`
`  (Contexto) contexto,`
`  (double) tolerancia`
`);`
	`(void) sai(`
	`);`

Busca por contexto	**Monitoramento de contexto**

`(void) busca(`
`  (Contexto) contexto,`
`  (double) tolerancia`
`);`
	`(void) inscreve(`
	`  (IP) monitorado,`
	`  (Time) intervalo`
	`  (Contexto) contexto`
	`);`

determina quão parecido será o contexto do nó retornado pela busca em relação ao contexto procurado. No caso da adesão, quão similares deverão ser os nós de um *cluster* em relação ao contexto do nó que irá ingressá-lo. A tolerância é expressa em termos percentuais, ou seja, o percentual mínimo de semelhança desses contextos.

As figuras de 4.3 a 4.6 representam os diagramas de sequência de cada um dos métodos da interface programática da arquitetura. A adesão envolve um processo decisório, representado em seu diagrama de sequência pelo quadro no qual um dos fluxos apresenta o caso em que o nó adere a determinado cluster e o outro fluxo mostra o caso em que esse nó se torna líder de um novo cluster.

O método de saída da arquitetura apresenta dois comportamentos possíveis, representados em seu diagrama de sequência. No primeiro caso, o nó que está deixando o cluster recebe uma confirmação do líder de seu cluster, reconhecendo sua saída. No segundo caso, o nó que deixa o cluster não aguarda essa confirmação de seu líder, optando por sair do cluster de forma não graciosa. Para lidar com esses casos, os líderes de clusters devem, com intervalos pré-definidos, recalcular o centróide do cluster, consultando os membros garantindo que a medida do centróide seja atualizada em caso de falhas ou saídas não graciosas de nós.

O método de busca primeiro consulta os líderes de clusters optando por enviar uma mensagem de busca apenas para aquele cujo centróide é mais semelhante ao contexto procurado. Essa escolha pode implicar em um desempenho pior das buscas, uma vez que o participante ou aqueles participantes cujos contextos satisfazem a *query* realizada, podem estar associados a um cluster diferente daquele escolhido. No teste realizado, essa degradação dos resultados foi de no máximo 30% e em mais de 55% dos

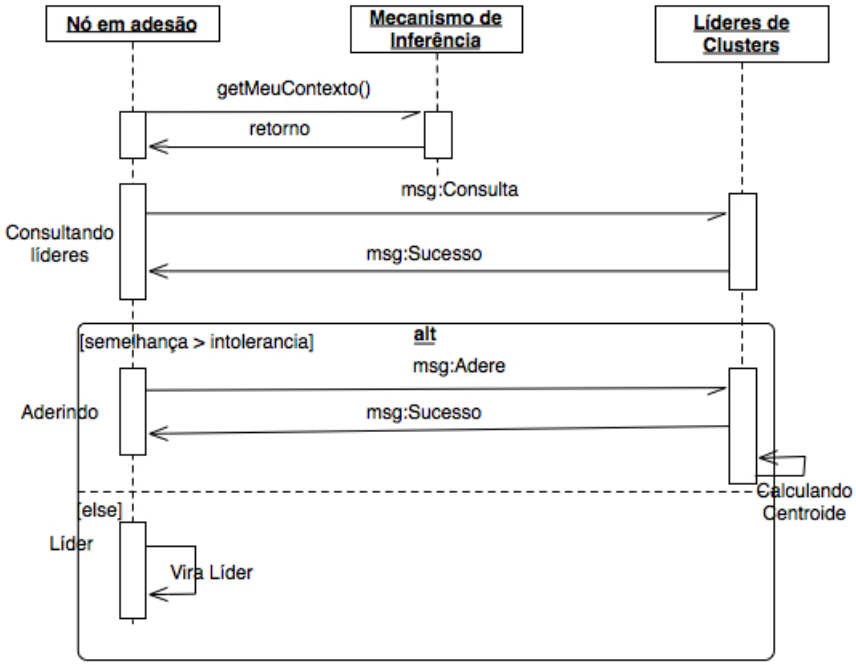


Figura 4.3. Diagrama de sequência da Adesão

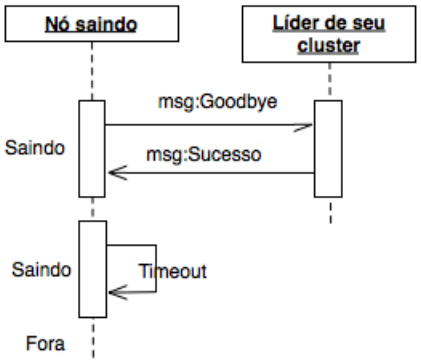


Figura 4.4. Diagrama de sequência da Saída

testes a busca retornou o indivíduo ideal. Esses resultados serão discutidos em mais detalhe no capítulo 5.

O método de monitoramento de contexto funciona através da interpelação periódica do usuário cujo contexto está sendo monitorado por parte do líder de seu cluster. Eventualmente, quando o usuário monitorado apresentar o contexto estipulado pelo participante que o monitora, encerra-se o funcionamento do método de forma satisfatória.

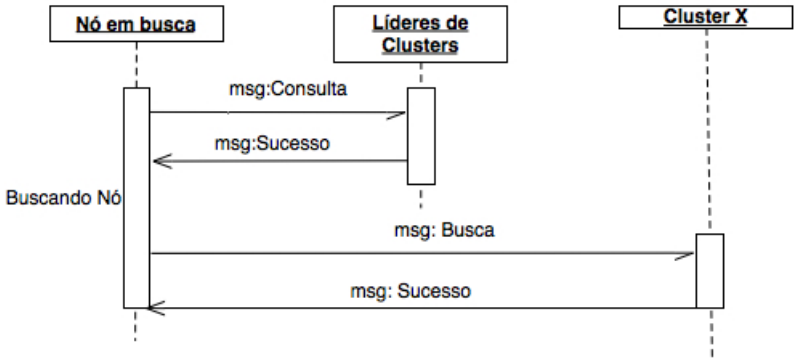


Figura 4.5. Diagrama de sequência da Busca

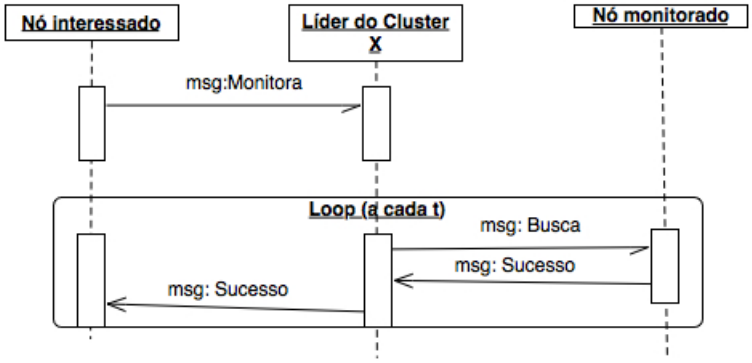


Figura 4.6. Diagrama de sequência do Monitoramento de Contexto

4.6 Aplicações sociais móveis

Para compreender a utilidade da arquitetura é interessante definir as características das aplicações que poderiam se beneficiar do uso da mesma. A primeira dessas características é a mobilidade. Por se tratar de uma arquitetura que implementa um protocolo e uma camada *overlay* cientes de contexto, aplicações móveis se beneficiariam principalmente pelo fato do contexto ser um objeto central na arquitetura, influenciando na sua organização, nos resultados das buscas e no monitoramento.

Outras aplicações para as quais a arquitetura é interessante são aquelas cuja comunicação tem um aspecto social. A clusterização é uma tendência das redes sociais. O mecanismo de organização da camada *overlay peer-to-peer* foi inspirado nessa característica e tenta organizar os nós de forma semelhante àquela que seria a rede social real da aplicação funcionando sobre a arquitetura. Em função dessa organização, as aplicações sociais se beneficiariam do melhor desempenho nas buscas por usuários e da proximidade topológica entre usuários relacionados socialmente.

No entanto, existe um paradoxo relativo a semelhança entre a topologia real da rede social e a topologia da camada *overlay*. Uma infraestrutura mais semelhante à rede social implica em melhor desempenho na comunicação entre usuários com relacionamentos sociais estabelecidos, porém prejudica o desempenho da busca por usuários localizados fora do cluster do usuário, em função do aumento do número de clusters. A utilização de um número de clusters menor do que aquele encontrado na rede social real, prejudica o desempenho das comunicações entre usuários relacionados entre si, mas otimiza o desempenho das buscas por usuários localizados em outros clusters. Existe um parâmetro na arquitetura denominado `tolerância`, citado também na seção anterior, que determina a suscetibilidade dos usuários a aderir a determinado grupo, mesmo que ali dentro existam pessoas com quem ele não tenha uma relação social. Quanto menor essa tolerância, maior o número de clusters, o que é recomendado para aplicações nas quais a ocorrência de comunicação entre usuários que não possuem uma relação social, previamente estabelecida, é mais rara. Por outro lado, quanto maior a tolerância, maior o tamanho dos clusters, o que é recomendado para aplicações na qual a formação de novos relacionamentos é mais comum.

A `tolerância` é um parâmetro que o desenvolvedor deve definir ao implementar sua aplicação utilizando a arquitetura deste trabalho. Além dela, ele deve definir o modelo lógico que descreve o conhecimento do problema e implementar os métodos de classes abstratas definidas na arquitetura. Para explicar a metodologia de desenvolvimento associada ao uso da arquitetura para desenvolver aplicações sociais móveis, é interessante utilizar como exemplo os passos necessários para implementar a aplicação concebida para este trabalho, chamada de *Social Campi*. Ela foi projetada de forma a englobar as características desejáveis para as aplicações suportadas pela arquitetura deste trabalho.

4.6.1 *Social Campi*

O primeiro passo para o desenvolvimento de uma aplicação social móvel utilizando esta arquitetura, é determinar quais informações irão compor o contexto e implementar uma forma de coletar essas informações. Essa coleta pode ocorrer das seguintes formas: atuação do usuário - descrevendo suas preferências e outras informações pessoais; coleta de sensores individuais – presentes em celulares, carros ou outras soluções de computação vestível (*wearable computing*), como medidores de pressão sanguínea; sensores coletivos – implantados nos ambientes dos usuários através de câmeras, transceptores sem fio (e.g., Bluetooth) ou ainda a coleta de rastros de sinais de celular.

A coleta de informações de contexto não faz parte do escopo desse trabalho,

Tabela 4.4. Informações de associação na base de conhecimento

```
associado('AdmBldg16AP1','00a0f8ab070d',1026840585,1026842553).
associado('LibBldg2AP7','00a0f8ab070d',1026842553,1026842579).
```

principalmente pela especificidade da implementação. Assume-se que cada usuário já possui as informações de seu contexto armazenadas na forma de variáveis numéricas e predicados lógicos que avaliam essas variáveis. Para demonstrar que é viável utilizar dados reais de contexto para implementar uma aplicação utilizando a arquitetura proposta, optou-se por utilizar um conjunto de dados com informações de associação de usuários a pontos de acesso sem fio no campus da universidade de Dartmouth em New Hampshire, US [Kotz et al., 2009]. A motivação para a escolha desse conjunto de dados é a sua adequação à simulação de aplicações sociais móveis demonstrada principalmente em Vaz de Melo [2011]. Nesse trabalho, o autor desenvolve um classificador de relacionamentos na rede social, que os diferencia em quatro tipos distintos: amigos, conhecidos, pontes e aleatórios. Além do algoritmo de classificação, o trabalho supracitado demonstra o caráter social das informações disponíveis no conjunto de dados. Por essa razão, optou-se por utilizar esse mesmo conjunto de dados para simular o uso de uma aplicação social pelos usuários cujas características estão ali representadas.

O conjunto de dados escolhido apresenta arquivos com os horários de associação e desassociação de cada usuário aos pontos de acesso de redes sem fio da universidade. Portanto, o primeiro passo para a inserção desse conhecimento no mecanismo de inferência é a inclusão desses fatos na base de conhecimento. Os fatos representados na tabela 4.4 são exemplos dentre aqueles que foram inseridos na base de conhecimento. O primeiro parâmetro representa o nome do ponto de acesso, que segue uma lógica que informa qual o seu prédio de instalação e andar. O segundo identifica o usuário através do endereço físico da sua placa de rede sem fio. Ressalta-se que não são considerados casos nos quais um dispositivo móvel possa ter mais de uma placa de rede sem fio e tampouco casos nos quais o usuário altere o endereço físico de sua placa para se passar por outra pessoa. Os últimos dois parâmetros representam, respectivamente, os horários de associação e de desassociação dos usuários àquele ponto de acesso, representados em segundos desde 1º de Janeiro de 1970 (*Unix Epoch Time*). A partir desses fatos é possível obter respostas do mecanismo de inferência à perguntas como "onde estava fulano às 8 da noite de determinado dia" ou "quem estava na sala 7 do segundo andar da biblioteca nesse mesmo horário".

No caso da aplicação *Social Campi*, o contexto de cada usuário será representado

pela frequência com a qual esse usuário esteve associado a cada ponto de acesso. Por essa razão, foram implementados os predicados `tempo` e `frequencia` representados na tabela 4.5. A distribuição das frequências é uma representação da mobilidade de um usuário. A figura 4.7 mostra a distribuição de frequências do usuário '0000c5112528' em função dos pontos de acesso aos quais esteve conectado em algum momento.

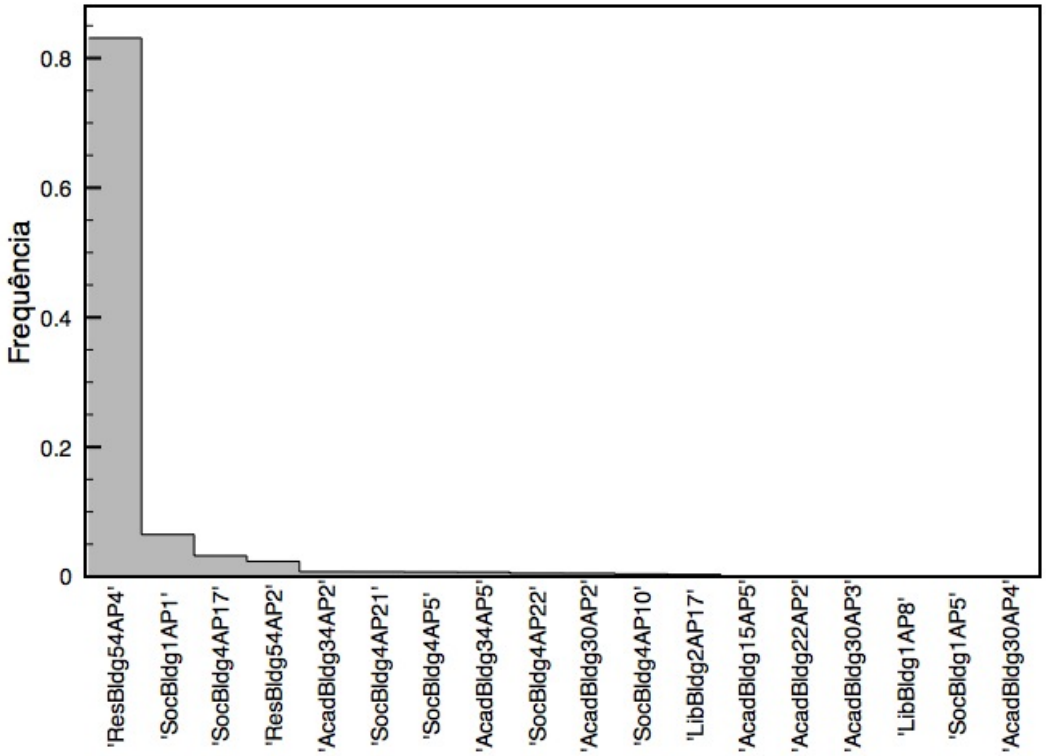


Figura 4.7. Distribuição de frequências do usuário '0000c5112528'

Dois usuários com a mobilidade semelhante têm maior probabilidade de interagirem, pois frequentam os mesmos lugares. Por essa razão, a clusterização dos usuários é feita em função da semelhança entre as frequências dos usuários. Essa semelhança é calculada utilizando a fórmula de distância geométrica, também conhecida como soma dos quadrados.

$$D = \sqrt{(f_{ai} - f_{bi})^2 + f_{aj} - f_{bj})^2 + \dots} \tag{4.2}$$

f_{ai} é a frequência do tempo em que o usuário a este conectado a um ponto de acesso i.

Entretanto, a clusterização usando apenas a semelhança entre as frequências produz um número muito grande de clusters, o que deterioraria a eficácia das buscas por conteúdo na arquitetura. A solução para esse problema foi avaliar também, durante o processo de decisão sobre qual cluster aderir, a quantidade de usuários de cada cluster

Tabela 4.5. Predicados para representação do contexto

```
tempo(AP, User, X) :- bagof([FIM,INI],
                       associado(AP, User, INI, FIM), Bag),
                       sum(Bag, X).
tempoTotal(User, TOT) :- bagof([AP,F],
                          tempo(AP, User, F), Bag),
                          sumFreq(Bag,TOT).
frequencia(AP, User, FREQ) :- tempo(AP, User, X),
                              tempoTotal(User, TOT),
                              FREQ is X/TOT.
sum(List,X):-sum(List,0,X).
sum([[FIM,INI] | T],X,Y):-Z is FIM-INI+X, sum(T,Z,Y).
sum([],X,X).
sumFreq(List,X) :- sumFreq(List,0,X).
sumFreq([[AP,TEMPO]T],X,Y) :- Z is TEMPO+X, sumFreq(T,Z,Y).
sumFreq([],X,X).
```

com os quais o usuário que está aderindo já teve contato. Essa abordagem é semelhante àquela usada em Vaz de Melo [2011]. Nesse trabalho os autores definem duas métricas para distinguir relacionamentos sociais de relacionamentos aleatórios: a regularidade – relacionamentos sociais se repetem ao longo do tempo e a similaridade – é esperado que indivíduos que compartilhem um relacionamento social possuam conhecidos em comum. A utilização da regularidade e da similaridade entre os usuários proporciona uma clusterização mais adequada, na qual a grande maioria das relações de amizade identificadas em Vaz de Melo [2011] acontecem entre usuários de um mesmo cluster, como demonstrado na seção 5.2.

Para facilitar as buscas na aplicação *Social Campi*, também estão armazenados no contexto o endereço físico da placa de rede sem fio e o papel de cada usuário na universidade – professor, aluno ou funcionário administrativo. Para fins ilustrativos, são considerados professores todos os usuários que passam mais do que 75% do tempo nos prédios acadêmicos, alunos são todos os usuários que tenham frequentado os prédios de residências estudantis por mais do que 20% de seu tempo, e funcionários administrativos são aqueles que passam mais do que 75% de seu tempo nos prédios administrativos. Esses predicados estão representados na tabela 4.6. Para caracterizar os diferentes prédios nos quais estão localizados os pontos de acesso, foi necessário também criar alguns predicados que analisam o nome dos APs e determinam esses prédios segundo uma lógica de nomenclatura descrita pela documentação do conjunto de dados [Kotz et al.,

Tabela 4.6. Funções de usuários e caracterização dos APs

```
administrativo(AP) :- contains(AP, "Adm").
residencial(AP) :- contains(AP, "Res").
academico(AP) :- contains(AP, "Acad").
atleticos(AP) :- contains(AP, "Athl").
sociais(AP) :- contains(AP, "Soc").
bibliotecas(AP) :- contains(AP, "Lib").

professor(User) :- associado(AP, User, INI, FIM),
                   academico(AP), bagof([AP,FREQ],
                   frequencia(AP, User, FREQ), Bag),
                   sumFreq(Bag,F), F > 0.75.
aluno(User) :- associado(AP, User, INI, FIM),
               residencial(AP), bagof([AP,FREQ],
               frequencia(AP, User, FREQ), Bag),
               sumFreq(Bag,F), F > 0.20.
funcionario(User) :- associado(AP, User, INI, FIM),
                     administrativo(AP), bagof([AP,FREQ],
                     frequencia(AP, User, FREQ), Bag),
                     sumFreq(Bag,F), F > 0.75.
```

2009]. A diferenciação de funções entre os usuários a partir das frequências de associação aos pontos de acesso é ilustrativa e poderia ser fornecida pela própria universidade, de forma mais confiável.

Provavelmente, nos casos de outras aplicações mais realistas, nas quais a coleta do contexto é distribuída, a informação armazenada em cada nó é mais complexa e pode incluir preferências pessoais, informações históricas e valores obtidos por sensores em tempo real.

Para poder extrair mais conhecimento a partir desses fatos, foram definidas algumas relações representadas na tabela 4.7. O predicado `presente` permite determinar se um usuário esteve presente em algum lugar entre um horário de início e fim e o predicado `presentes` informa quais usuários estavam naquele local. Os predicados `locais_de_encontro` e `num_encontros` permitem determinar, respectivamente, onde determinados usuários se encontraram e em quantos locais diferentes isso ocorreu. A decisão sobre quais predicados implementar no modelo fica a cargo do desenvolvedor, porém os predicados que irão constituir o centróide dos clusters devem ser escolhidos objetivando a otimização das buscas. São esses predicados que irão determinar a probabilidade de sucesso das buscas na camada *overlay*. Para o caso da aplicação *Social*

Tabela 4.7. Predicados para extração do conhecimento

```
presente(AP, User, INI, FIM) :- associado(AP, User, X, Y),
                                X < FIM, X > INI;
                                associado(AP, User, X, Y),
                                Y > INI, Y < FIM.
presentes(AP, Users, INI, FIM) :- setof(User,
                                  presente(AP, User, INI, FIM),
                                  Users).
locais_de_encontro(AP, User1, User2) :- associado(AP, User1, X, Y),
                                   associado(AP, User2, Z, W),
                                   X < Z, Y > Z;
                                   associado(AP, User1, X, Y),
                                   associado(AP, User2, Z, W),
                                   X > Z, Y < W.
num_encontros(User1, User2, X) :- setof(AP,
                                  locais_de_encontro(AP,
                                  User1, User2), APs),
                                  length(APs, X).
```

Campi, optou-se por calcular o centróide a partir da média das frequências com as quais os participantes daquele cluster visitaram cada ponto de acesso, por essa razão os predicados de cálculo dessa são importantes para a aplicação. Os outros predicados são implementados para permitir a extração de conhecimento mais estruturado, utilizado pela busca e monitoramento de contexto da aplicação *Social Campi*. Os exemplos desse uso estão descritos mais adiante nesta seção.

Embora ainda seja relativamente simples, o modelo lógico da tabela 4.7 demonstra a força de representação da linguagem Prolog, através da recursão e enumeração de combinações possíveis (`bagof`, `setof`). Essa representatividade é particularmente importante, pois torna a coleta do contexto mais simples, como é o caso do conjunto de dados utilizado nesse trabalho, a partir do qual foi possível extrair muita informação sem a necessidade de aumentar a complexidade da coleta ou da representação dos fatos na base de conhecimento. Para a melhor compreensão da linguagem Prolog, consulte Clocksin & Mellish [2003].

Após a definição das informações que irão compor o contexto, a implementação dos predicados para extração do conhecimento e a definição das informações que irão compor os centróides, é possível iniciar a construção da camada *overlay* para o uso pela aplicação. Antes disso, no entanto, é necessário sobrecarregar o método

`escolheCluster(lista_de_clusters, tolerancia)`, que deve retornar o endereço do líder do cluster ao qual cada usuário deve aderir ou seu próprio endereço IP, caso opte por criar um novo cluster. No caso da aplicação *Social Campi*, para cada adesão o novo participante avalia a distribuição de frequências dos clusters existentes, para determinar se existe algum deles que se assemelha às frequências com as quais ele se associa a cada um dos pontos de acesso, e ainda avalia a sua afinidade com os membros daquele cluster, em termos do número de membros com os quais já se encontrou. À cada adesão ou saída a arquitetura realiza uma troca de mensagens para recalcular os centróides influenciados pela mudança na topologia. Os tempos e números de mensagens desse processo de estabilização estão demonstrados no capítulo 5.

A estrutura da camada *overlay* construída está em constante mutação, em função das alterações de contexto recorrentes dos usuários de dispositivos móveis, e a consequente troca de clusters, em função dessas alterações. Entretanto, após a adesão de todos os participantes, representados no conjunto de dados dos usuários no campus de Dartmouth, considera-se que a construção da camada *overlay* está completa. Principalmente a partir desse momento, mas também durante o processo de adesão, é possível executar as seguintes operações de comunicação, implementadas com fins ilustrativos:

- Avisar aos meus colegas da disciplina MAT001: quem conseguiu fazer o exercício 3 da lista?
- Avisar aos meus alunos da disciplina BIO002: prova remarcada para o dia 05/05.
- Ser informado quando determinado professor estiver no prédio Acadêmico.
- Avisar aos alunos residentes no bloco 83, oitavo andar: alguém perdeu um sapato no corredor.

A escolha desses exemplos tem a finalidade de explorar as funcionalidades de busca e monitoramento de contexto, proporcionadas pela arquitetura, e demonstrar a capacidade de extração de conhecimento dos predicados. As buscas são fictícias, portanto para fins ilustrativos, supõem-se que a disciplina MAT001 é lecionada na sala de aula onde está localizado o ponto de acesso "AcadBldg6AP4", e BIO002 é lecionada na sala de aula onde está o AP "AcadBldg6AP2". O próximo capítulo (5), demonstra a escalabilidade da arquitetura proposta através de simulações e utiliza os exemplos de busca definidos aqui para demonstrar o funcionamento da aplicação *Social Campi*. Essa aplicação é um exemplo real da viabilidade do uso da arquitetura proposta para desenvolver aplicações sociais móveis.

Capítulo 5

Simulação

A simulação de redes p2p é um tema amplamente abordado pelas áreas de algoritmos distribuídos [McCanne et al., 1997; Baumgart et al., 2007; Pongor, 1993; Montresor & Jelasity, 2009], e computação móvel [Wang et al., 2003; Piorkowski et al., 2008], apesar de existirem poucos esforços direcionados à computação ubíqua e à adaptação ao contexto [Barton & Vijayaraghavan, 2002]. A maior vantagem de se utilizar um simulador já consolidado é a confiabilidade dos algoritmos e protocolos utilizados pelo mesmo. Ao utilizar esses simuladores, o pesquisador diminui a probabilidade de que os resultados de seus experimentos sejam impactados por diferenças nas implementações dos protocolos, uma vez que a implementação dos mesmos já foi testada em diversas outras iniciativas de pesquisa.

Entretanto, a especificidade dos algoritmos desenvolvidos neste trabalho, todos com foco no contexto, sua busca, comparação ou clusterização, inviabiliza a adaptação de protocolos existentes, reduzindo o ganho no uso de simuladores pré-existentes. Além disso, a complexidade no uso desses simuladores é normalmente compensada por funcionalidades úteis como a simulação de perdas de pacotes, ruídos, interferências, gargalos e limitações de banda, que não constituem aspectos a serem testados neste trabalho, que é focado principalmente na escalabilidade e aplicabilidade da arquitetura proposta.

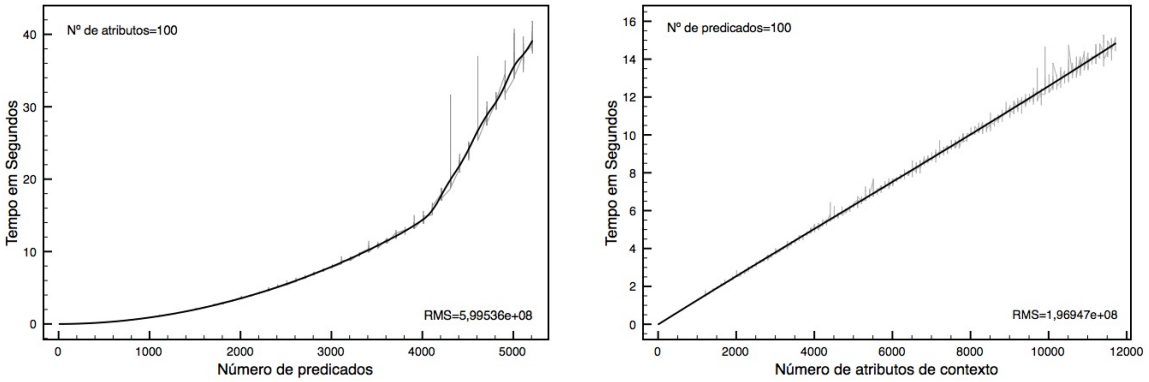
Optou-se, portanto, por implementar um simulador discreto de eventos, que apresenta mecanismos para avaliação da escalabilidade dos algoritmos, como tempos de execução com precisão de nanosegundos, estatísticas de números de mensagens, carga individual de nós participantes e carga total na rede. As medições dos experimentos foram realizadas trinta e três vezes, as amostras estão representadas nos gráficos através das linhas de cor mais clara. A partir dessas amostras realizou-se uma interpolação *spline*, utilizando polinômios de graus até três, para cada intervalo de interpolação.

A interpolação *spline* é mais simples do que a interpolação polinomial, possui uma aproximação satisfatória [Walsh et al., 1962] e implementação publicamente disponível, descrita em Thijsse [2002]. O propósito desse tipo de interpolação é representar a tendência de uma amostra, filtrando o ruído das medições. Esse ruído é quantificado através da média quadrática ou a raiz quadrada da média aritmética dos quadrados (*root mean square*) das amostras e está registrado no canto inferior direito dos gráficos, sempre na unidade do eixo y.

5.1 Escalabilidade

O simulador insere a aleatoriedade necessária aos experimentos através de um algoritmo de geração de modelos lógicos hipotéticos, variando-os de modelos mais simples, com poucos predicados, até modelos capazes de representar problemas mais complexos. O algoritmo 1 descreve, em pseudo-código, o processo de geração dos modelos lógicos. Embora não constitua parte da solução proposta para o desenvolvimento de aplicações sociais móveis, o custo da execução desse algoritmo de geração de modelos lógicos limita a magnitude dos experimentos, em termos do número de predicados e de atributos de contexto utilizados. A figura 5.1 mostra os tempos de execução desse algoritmo em função do número de predicados do modelo e em função do número de atributos representáveis no contexto de cada usuário.

Figura 5.1. Criação do modelo lógico hipotético



No uso real da arquitetura, o ônus de lidar com um número grande de predicados é drasticamente reduzido, uma vez que os participantes da arquitetura não precisam armazenar as regras de inferência, bastando consultar as entidades responsáveis pela inferência do contexto. No caso deste trabalho, o mecanismo de inferência foi im-

Algoritmo 1: Geração de modelos lógicos

Entrada: número de predicados: $p \geq 0$, número de atributos: $v \geq 0$, probabilidade $\alpha \mid 0 < \alpha < 1$
Saída: modelo: $M(P, V)$
para $i = 0$ *to* p **faça**
  $strings[i] = random(String[8])$;
para $i = 0$ *to* p **faça**
  para $j = 0$ *to* $j + i + 1 < p$ **faça**
    $x = rand(0 : 1)$;
    se $x > \alpha$ **então**
      $P(i) += strings(j)$;
    $x = 0 \vee 1$;
    se $x = 0$ **então**
      $P(i) += \wedge$;
    senão
      $P(i) += \vee$;
  $x = [0 \vee 1 \vee \ldots \vee v - 1]$;
  $P(i) += V(x)$;
retorna $M(P, V)$;

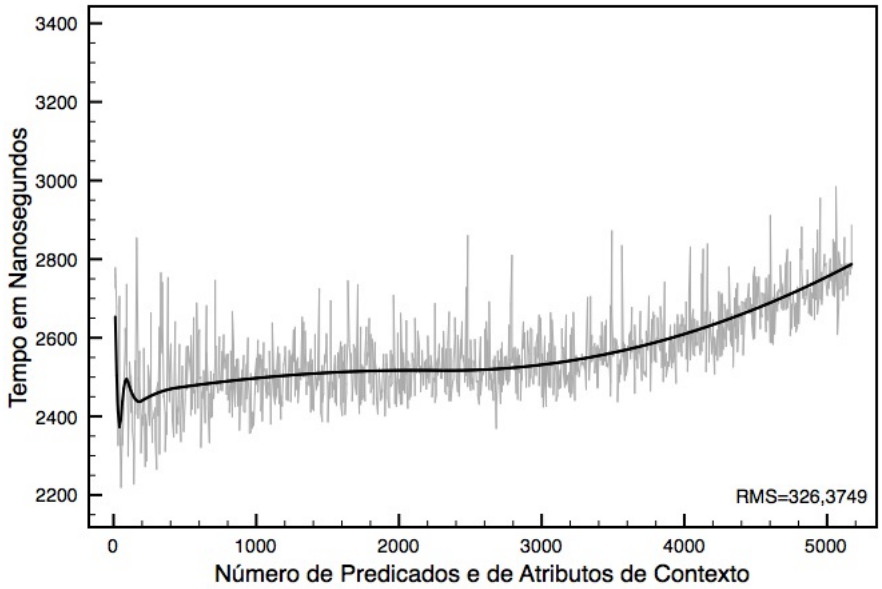
plementado em Prolog como uma entidade separada dos participantes, que pode ser consultada através de uma interface de *webservices*.

A figura 5.2 demonstra o crescimento nos tempos de inferência dos predicados à medida que o número destes aumenta. A análise da complexidade do algoritmo de avaliação de predicados do Prolog está fora do escopo deste trabalho, para maiores informações, consultar Harvey [1995]. Entretanto, o gráfico demonstra que, para modelos lógicos gerados aleatoriamente segundo o pseudo-algoritmo 1, os tempos de inferência apresentam um crescimento sublinear em relação ao número de predicados, permitindo a utilização de modelos lógicos suficientemente grandes para a maioria dos problemas.

5.1.1 Criação da Rede

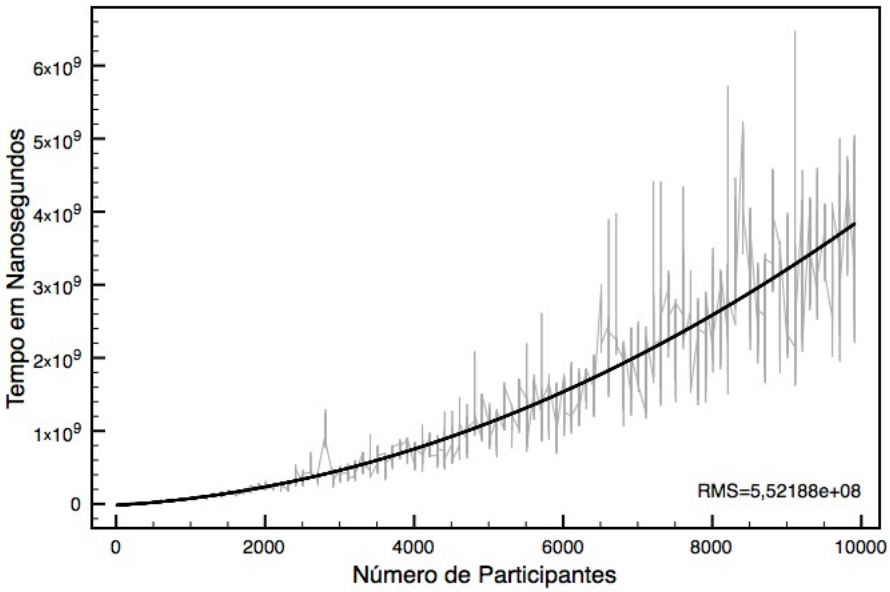
Embora individualmente cada consulta à entidade de inferência de contexto não inviabilize a arquitetura, durante o estágio inicial de organização dos nós, essas consultas acontecem de forma mais recorrente e, por essa razão, é necessário avaliar o desempenho da arquitetura no estágio inicial de múltiplas adesões de nós. A figura 5.3 demonstra o crescimento no tempo em função do número de participantes. Existe uma correlação entre o número de participantes e o tempo necessário para a adesão de todos eles à camada *overlay*, porém o tempo necessário cresce super-linearmente em função do nú-

Figura 5.2. Inferência de predicados em função do número de atributos e predicados



mero de participantes. Para entender esse comportamento, é necessário compreender o processo decisório pelo qual cada nó passa, individualmente, descrito pelo algoritmo 2.

Figura 5.3. Tempo para construção inicial da rede em função do número de participantes



Algoritmo 2: Processo decisório de adesão de um participante

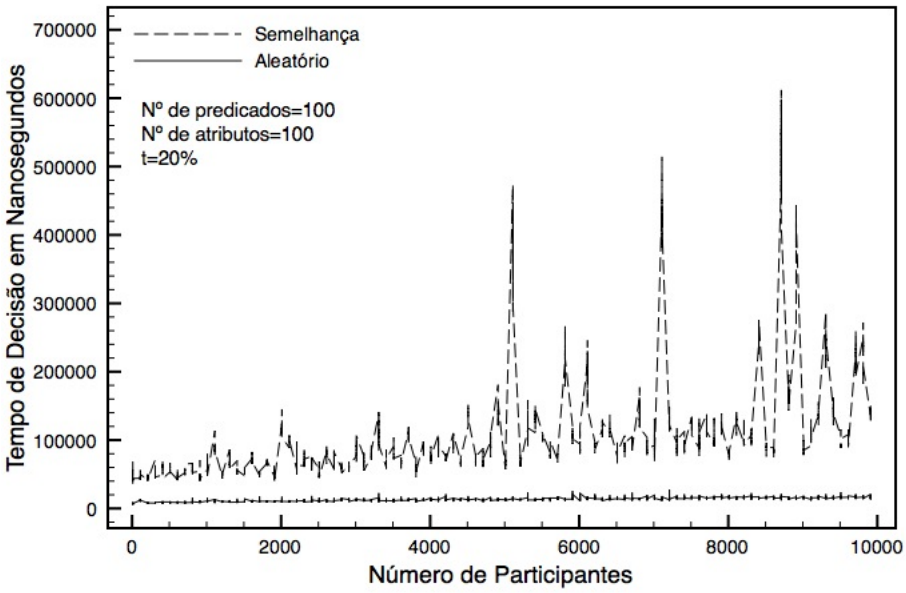
Entrada: atributos: $A(N)$, clusters: $G(V)$, predicados: $p_i(x)$, tolerância: t
Saída: cluster: $g \in G(V, E)$
para $i = 0$ *to* p **faça**
  $contexto(i) = p_i(A(N))$;
para $j = 0$ *to* V **faça**
  $X(j) = similaridade(contexto, centroide(j))$;
se $max(X(j)) \leq t$ **então**
  $g = i$
senão
  $g = V + 1$
retorna $M(P, V)$;

Primeiramente, cada participante fornece seus atributos $A(N)$ ao interpretador de contextos, que retorna o contexto referente àquele participante. Em seguida, o participante o compara com o centróide de cada cluster, para obter aquele com centróide mais semelhante ao seu contexto. Finalmente, ele opta por aderir ou não àquele cluster, baseado no parâmetro de tolerância, tornando-se líder de um novo cluster, no último caso. A figura 5.4 mostra o tempo médio gasto por cada participante para decidir a qual cluster irá aderir, utilizando o critério de semelhança entre o seu contexto e o centróide dos clusters ou optando aleatoriamente. À medida que o número de participantes aumenta, também cresce o número de clusters. No momento de sua adesão, cada participante deve avaliar cada um dos clusters formados previamente, para definir a qual irá aderir. Isso implica em um crescimento no tempo de decisão de cada participante individualmente, demonstrado na figura 5.4, que quando avaliado do ponto de vista da construção de toda a rede, gera um crescimento super-linear no tempo necessário para a adesão de todos os participantes.

O número de clusters formados na rede é influenciado pela tolerância dos participantes em relação a assimilaridade de contextos dos participantes do cluster ao qual decidiram aderir. Esse parâmetro t é definido globalmente para todos os participantes da camada *overlay*, mas nada impede que seja particular de cada participante, em função de uma menor ou maior predisposição pessoal à agrupar-se com usuários menos similares. Se definido globalmente, esse parâmetro influencia diretamente no número de clusters formados, tal que $\lim_{t\to 0} c(t) = n$ e $\lim_{t\to 1} c(t) = 1$, onde $c(t)$ é o número de clusters e n o número de participantes. Esses limites podem ser constatados na figura 5.5.

O número de predicados do modelo lógico também influencia no número de clusters formados. O número máximo de contextos possíveis é uma permutação dos pre-

Figura 5.4. Tempo para escolha de *cluster* na adesão



dicados. Dois ou mais participantes de contextos idênticos irão sempre pertencer a um mesmo cluster. Consequentemente, o número de clusters dependerá do número de contextos possíveis. A figura 5.6 mostra a variação do número de clusters em função do número de predicados do modelo lógico.

Figura 5.5. Variação do número de clusters em função da tolerância e do número de participantes

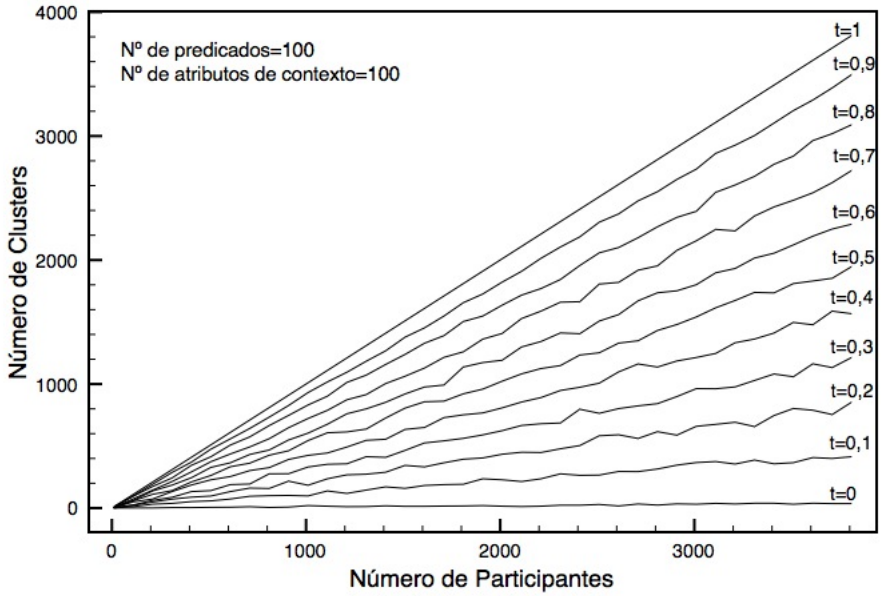
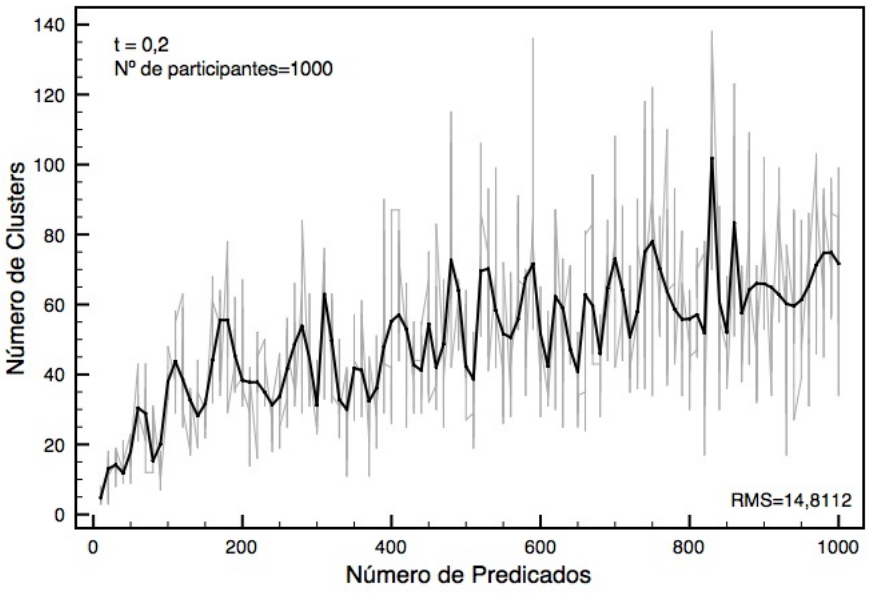


Figura 5.6. Número de *clusters* em função do número de predicados



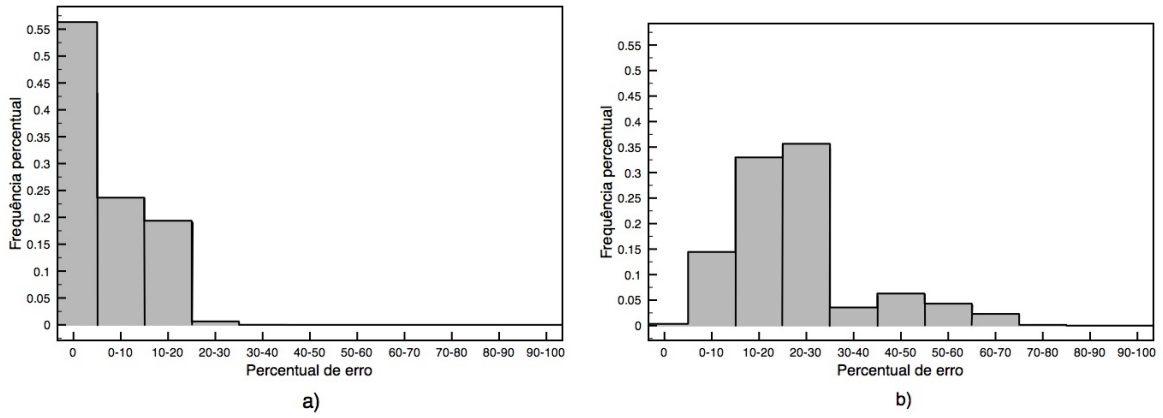
A figura 5.4 mostra que existe um ônus associado à utilização da semelhança entre contextos para organizar os participantes da arquitetura, se comparado ao processo decisório aleatório. No entanto, essa forma de organização beneficia o desempenho das buscas, tanto em tempo quanto em número de mensagens transmitidas na rede. Realizar uma busca significa localizar o indivíduo, ou o conjunto de indivíduos, cujo contexto mais se assemelha àquele que se deseja encontrar. No caso de uma rede não estruturada, essa busca é feita através de *flooding* e, embora retorne sempre o melhor resultado, incorre em um número maior de mensagens trocadas na rede. Por outro lado, em uma rede estruturada e, no caso desta arquitetura, clusterizada de forma ciente de contexto, as buscas são executadas com um número menor de mensagens, na média, embora possam retornar indivíduos menos adequados ou até nenhum indivíduo. A figura 5.7 apresenta histogramas do percentual de diferença entre o contexto procurado e o melhor resultado encontrado pelas buscas, realizadas em duas camadas *overlays* construídas de forma diferente. O gráfico a) representa as buscas em uma camada *overlay* clusterizada usando a similaridade entre os contextos, enquanto o gráfico b) representa os percentuais de erro de buscas executadas em uma rede estruturada de forma aleatória. Estes histogramas foram gerados a partir de mil execuções de buscas, nas quais se escolhia um participante aleatório da arquitetura e disparava-se uma busca utilizando o seu contexto.

Apesar de alguns resultados das buscas aproximadas não retornarem o melhor indivíduo para aquela busca, aqueles retornados apresentam uma relativa similaridade

com o contexto almejado, podendo, em muitos casos, serem utilizados sem prejuízo para a aplicação. Além disso, as aplicações que utilizam esse tipo de buscas, normalmente o fazem para encontrar um número X de indivíduos mais semelhantes ao contexto almejado, sem o requisito de que todos sejam idênticos.

Em contrapartida à perda relativa na qualidade de algumas buscas, a arquitetura proposta proporciona a redução no número de mensagens que atravessam a rede. A figura 5.8 mostra a média do número de mensagens necessárias para as buscas em função do aumento do número de indivíduos no gráfico à esquerda, e compara o número essa média com a quantidade de mensagens necessárias se utilizado um algoritmo de *Flooding* ou o protocolo *Chord* [Stoica et al., 2001], um protocolo eficiente para busca de em redes P2P. A representação do número de mensagens utilizados pelo protocolo *Chord* é ilustrativa, pois a próxima seção explica a impropriedade de se utilizar este ou outros protocolos P2P, notoriamente eficientes na busca de conteúdo, para o caso específico do problema tratado neste trabalho.

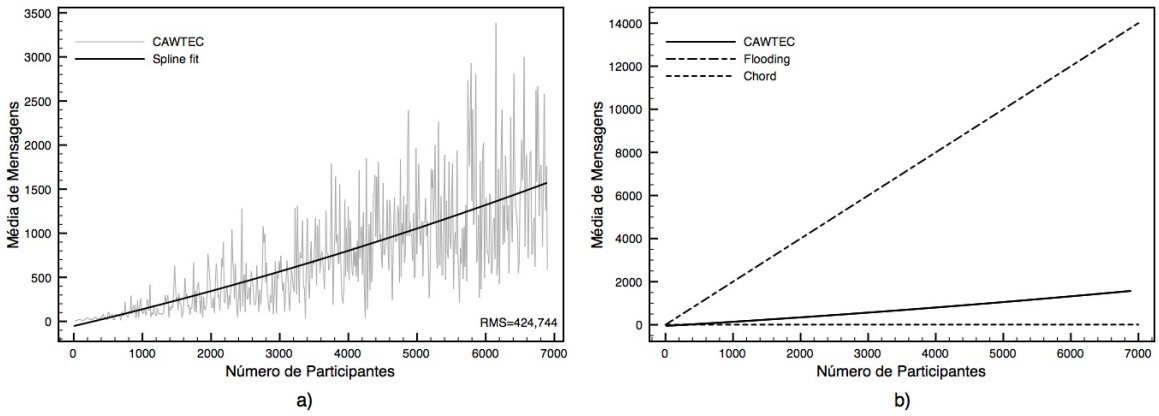
Figura 5.7. Efetividade das buscas: CAWTEC x rede estruturada aleatoriamente



5.2 Viabilidade

Embora representem um dos pilares da arquitetura para aplicações sociais móveis cientes de contexto, as buscas por contexto, por si só, poderiam ser implementadas de formas mais eficientes, principalmente através de protocolos *peer-to-peer* como o *Chord* [Stoica et al., 2001] ou o *Pastry* [Rowstron & Druschel, 2001], ambos com crescimento logaritmo do número de mensagens, no caso médio. Para compreender a razão pela qual esses protocolos são impróprios para arquiteturas para aplicações sociais móveis, e exemplificar o uso da arquitetura proposta neste trabalho, criou-se um protótipo

Figura 5.8. Média do número de mensagens em função do número de participantes



funcional de uma aplicação social móvel real, a partir de um *dataset* disponível publicamente. Esse *dataset* contém os rastros de mobilidade de 1146 usuários frequentadores do campus da Universidade de Dartmouth durante três anos [Kotz et al., 2009].

A seção 4.6.1 apresentou alguns exemplos de funcionalidades possíveis da aplicação *Social campi*. Não coincidentemente, todos os avisos citados de exemplo presumem que seja possível encontrar um número determinado ou indeterminado de usuários cujo contexto mais se assemelhe àquele procurado. É principalmente nesse quesito que os protocolos P2P, normalmente utilizados em soluções de busca de conteúdo, falham como solução para um protocolo P2P para aplicações sociais móveis.

Analogamente, podemos usar o problema da busca de documentos para exemplificar a mesma limitação dessas soluções. Os protocolos de busca P2P, dentre eles o *Chord* e o *Pastry*, atribuem chaves que correspondem aos documentos armazenados naquele nó participante da camada *overlay*, para futuramente serem obtidas de forma eficiente usando uma busca distribuída. Embora esses algoritmos de busca sejam eficientes e escaláveis, não existe uma forma de realizar uma busca aproximada, ou encontrar os n documentos mais apropriados à determinada consulta, sem modificar bastante os algoritmos originais [Shen et al., 2004]. A clusterização revela-se uma solução mais adequada às buscas aproximadas, uma vez que encurta as distâncias entre documentos semelhantes, e no caso das aplicações sociais móveis, entre usuários semelhantes e mais propícios a se comunicarem.

Para avaliar o desempenho da clusterização na arquitetura proposta neste trabalho, utilizou-se o resultado de um classificador de usuários, desenvolvido em Vaz de Melo [2011] e fornecido gentilmente pelo autor. Nesse trabalho, os autores utilizam o mesmo *dataset* com os rastros de mobilidade dos usuários do campus de *Dartmouth*

para desenvolver um avaliador de relações sociais. Ele utiliza dois critérios – a frequência com a qual dois usuários se encontram e a interseção de amigos que eles possuem – para classificar seu relacionamento em quatro níveis: amigos, conhecidos, pontes e aleatórios. A tabela 5.1 demonstra a eficiência da arquitetura em organizar os participantes priorizando a afinidade entre eles, pois cerca de 90% dos pares de participantes classificados como amigos por Vaz de Melo [2011] pertencem a um mesmo cluster na camada *overlay* construída pela arquitetura deste trabalho. Além disso, a tabela 5.2 mostra a distância média para cada um dos tipos de relacionamentos, quando utilizados clusters na forma de subgrafos completos.

Tabela 5.1. Desempenho da Clusterização

Tipo de Relacionamento	Total	% Intracluster	% Extracluster
amigos	31455	0.90964870449849	0.09035129550151
conhecidos	874	0.7883295194508	0.2116704805492
pontes	37795	0.85114433126075	0.14885566873925
aleatórios	23625	0.72156613756614	0.27843386243386

Tabela 5.2. Distância Média

Tipo de Relacionamento	Distância (em saltos)
amigos	1.793690
conhecidos	2.783110
pontes	2.281630
aleatórios	3.451669

Além de proporcionar uma camada *overlay* cuja topologia se assemelha à rede social formada pelos usuários que frequentam o campo de *Dartmouth*, a arquitetura desenvolvida neste trabalho permite a execução das buscas citadas na seção 4.6.1, detalhadas nas tabelas 5.3, 5.4, 5.5 e 5.6. Para o caso específico da busca representada na tabela 5.5, utilizou-se também o mecanismo de monitoramento de contextos e registrou-se o momento em que o primeiro retorno da chamada de procedimento ocorreu. Algumas suposições utilizadas nas buscas são provavelmente imprecisas e outras ilustrativas. Isso ocorre por que a extração do contexto da aplicação *Social Campi* está restrita aos rastros de mobilidade dos usuários, quando poderia obter informações a partir de registros de posse da própria universidade, como por exemplo quais as salas onde são lecionadas as disciplinas MAT001 e BIO002 e em qual horário. Entretanto, as buscas exemplificam possíveis funcionalidades de uma aplicação social móvel desenvolvida utilizando a arquitetura proposta neste trabalho.

Tabela 5.3. Busca 1: Avisar aos meus colegas da disciplina MAT001

Suposição:	MAT001 foi lecionada na sala "AcadBldg6AP4" de Thu, 25 Apr 2002 16:00:00 GMT à Thu, 25 Apr 2002 17:30:00 GMT
Código:	Inicio = EpochTime('Thu, 25 Apr 2002 16:00:00 GMT') Fim = EpochTime('Thu, 25 Apr 2002 17:30:00 GMT') Contexto = aluno(User) ∧ presente('AcadBldg6AP4', User, Inicio, Fim) Tolerancia = 0.9 busca(Contexto, Tolerancia)
Resultado	Users = {'00070e7040c6', '00070e96ba46', '00070e985250', '0007eba03671', '0040962a7d5e', '004096bcc4af', '004096c34f0d'}

Tabela 5.4. Busca 2: Avisar aos meus alunos da disciplina BIO002

Suposição:	BIO002 foi lecionada na sala "AcadBldg6AP2"' de Wed, 24 Apr 2002 16:00:00 GMT à Wed, 24 Apr 2002 17:30:00 GMT
Código:	Inicio = EpochTime('Wed, 24 Apr 2002 16:00:00 GMT') Fim = EpochTime('Wed, 24 Apr 2002 17:30:00 GMT') Contexto = aluno(User) ∧ presente('AcadBldg6AP2', User, Inicio, Fim) Tolerancia = 1 busca(Contexto, Tolerancia)
Resultado	Users = {'00070ea71ad6', '00070ec54833', '000750eb25de', '0007eba03671', '004096125cc4', '00409620f218', '00409629a622', '004096535f77', '00409665b7f1', '004096684347', '00409682f790'}

Tabela 5.5. Busca 3: Ser informado quando determinado professor estiver no prédio Acadêmico

Suposição:	Professor='0000c5e63857'
Código:	Agora = timeofday(); IP = IP(Professor); intervalo = 3600s; Contexto = associado(AP, Professor, Agora, Agora) ∧ academico(AP) inscreve(IP, Intervalo, Contexto)
Resultado	Retorna do callback quando agora = Wed, 09 May 2001 17:45:09 GMT

Tabela 5.6. Busca 4: Avisar aos alunos residentes no bloco 83, oitavo andar

Suposição:	Se um usuário passa mais de 80% de seu tempo de conexão, associado a determinado AP residencial, então é naquele andar que fica o seu quarto.
Código:	Contexto = frequencia('ResBldg83AP8', User, FREQ), FREQ $>$ 0.80. Tolerancia = 1 busca(Contexto, Tolerancia)
Resultado	Users = {'00053c247422', '00053c5bee98', '00053c6df20e', '00053c79bee3', '00053c8cec8d', '00053cd6bd0e', '00053ce13919', '00064b4372eb', '0020e0194d5d', '0020e03d5055', '0020e0884cef'}

Capítulo 6

Conclusão

Este trabalho e os experimentos registrados nele estabelecem o embasamento teórico para a implementação de uma arquitetura para desenvolvimento de aplicações sociais móveis. Tanto a análise da escalabilidade da arquitetura quanto a demonstração de sua viabilidade são necessárias, uma vez que individualmente não são suficientes para motivar a implementação da arquitetura proposta, mas em conjunto demonstram a possibilidade de implementar aplicações sociais móveis reais em uma arquitetura distribuída e escalável.

O contexto é um aspecto chave da arquitetura, principalmente se considerarmos que também compõem esse contexto as relações sociais entre os usuários da aplicação. É esse contexto que determina a topologia da camada *overlay*, permite que a aplicação localize usuários que atendam aos seus interesses e possibilita a adaptação às peculiaridades dos dispositivos móveis.

Existem aspectos que influenciam no desempenho da arquitetura proposta. Alguns deles inclusive não podem ser avaliados sem a utilização de dispositivos móveis reais e dispersos globalmente, como por exemplo o tempo de resposta das aplicações quando os participantes de um mesmo cluster estiverem em muitos países diferentes. Entretanto, a complexidade do contexto, em termos do número de predicados possíveis e número de variáveis, e o número de participantes da camada *overlay* alteram de várias formas os resultados obtidos utilizando a arquitetura. Ambos influenciam na topologia e no desempenho das buscas. Por outro lado, esse comportamento é esperado, uma vez que o contexto é avaliado tanto para determinar a topologia, quanto para direcionar as buscas e o número de participantes pode dificultar ou facilitar as buscas.

O objetivo da arquitetura proposta neste trabalho é auxiliar no desenvolvimento de aplicações sociais móveis, contribuindo para a obtenção de aplicações mais adequadas a esses dois paradigmas. Ela desonera o desenvolvedor de lidar com a localização de

usuários, embora ele ainda deva traduzir o contexto procurado na forma de expressões em Prolog. A arquitetura também proporciona uma topologia adequada para as interações sociais, agrupando os participantes em função da sua similaridade e interage de forma modular com um interpretador da linguagem Prolog, permitindo a inferência de contextos, restando ao desenvolvedor representar o conhecimento do domínio através de um modelo lógico.

Entretanto, ainda recaem sobre o usuário algumas responsabilidades que interferem na adequação da arquitetura para aplicações sociais móveis. A mais importante delas é a implementação da lógica e as funcionalidades da aplicação. Embora a arquitetura proporcione formas de embutir a ciência de contexto na lógica das aplicações, se o desenvolvedor não incluir esse contexto na lógica da aplicação, ela não será ciente de contexto. Portanto, a simples utilização da arquitetura implementada neste trabalho não é uma garantia de aplicações adaptadas aos ambientes móveis.

O mesmo pode-se dizer a respeito do caráter social das aplicações. A arquitetura proposta manipula a topologia da camada *overlay* com o intuito de aproximar usuários com maior probabilidade de interagirem socialmente. Se as interações através da aplicação não tiverem origem social ou ainda, se o fator motivante para essas interações não estiver expresso no contexto dos usuários, a topologia construída pela arquitetura não irá beneficiar o desenvolvimento daquela aplicação e a arquitetura se tornará um ônus. Para mitigar esse risco, é necessário que o desenvolvedor conheça os paradigmas de computação ubíqua e de redes sociais, compreenda os aspectos motivantes para as interações sociais na rede que deseja implementar e seja capaz de expressar esses aspectos através do modelo lógico e do contexto de cada usuário individualmente.

Embora o desenvolvedor ainda desempenhe um papel importante na concepção de boas aplicações sociais móveis, a arquitetura apresentada neste trabalho permite desenvolver aplicações nas quais o contexto é manipulado facilmente, através da interação com um interpretador lógico de contextos, a topologia é ciente de contexto e a proximidade social entre os usuários é utilizada para beneficiar as comunicações. Embora ainda existam diversos trabalhos futuros que podem auxiliar no desenvolvimento dessas aplicações, a arquitetura permite ao desenvolvedor concentrar-se mais na implementação da lógica da aplicação e dos requisitos funcionais de suas aplicações, contribuindo para o advento de aplicações sociais móveis mais úteis e mais adequadas para o uso cotidiano.

6.1 Trabalhos Futuros

Embora este trabalho apresente um embasamento teórico para a arquitetura proposta, ele apresenta algumas limitações, associadas tanto à comprovação da escalabilidade da arquitetura quanto à sua viabilidade. É necessário implementar o protocolo proposto nas plataformas móveis, disponibilizar essa implementação e documentá-la, para finalmente testar a escalabilidade e a viabilidade da arquitetura quando a camada *overlay* estiver sobreposta à Internet, com dispositivos dispersos globalmente.

A interface de métodos disponíveis na arquitetura, que conta apenas com os métodos de adesão e saída do *overlay* e busca e monitoramento de contextos, deve ser estendida para permitir maior usabilidade por parte dos desenvolvedores das aplicações.

Além disso, é interessante que sejam desenvolvidos trabalhos focados em capturar o mecanismo de formação de redes sociais reais, encontrar formas de expressá-lo através do contexto dos usuários e provar que, nesses casos, usuários com contextos semelhantes têm maior probabilidade de se relacionarem em uma rede social.

Apesar de existirem iniciativas em se padronizar a representação de contexto através de ontologias criadas para o domínio das aplicações móveis, é interessante que as relações interpessoais e outros aspectos sociais também possuam uma forma padronizada de representação e que exista uma iniciativa de criação de um interpretador de contextos compartilhado entre várias dessas aplicações, pois gradualmente as aplicações móveis se tornam cada vez mais sociais e as aplicações sociais cada vez mais móveis.

Referências Bibliográficas

Adamic, L. & Adar, E. (2005). How to search a social network. *Social Networks*, 27(3):187--203.

Adar, E. & Huberman, B. (2000). Free riding on gnutella. *First Monday*, 5(10-2).

Allam, S. (2009). Model to measure the maturity of smartphone security at software consultancies.

Allen, S.; Graupera, V. & Lundrigan, L. (2010). Titanium mobile. *Pro Smartphone Cross-Platform Development*, pp. 153--160.

Backstrom, L.; Huttenlocher, D.; Kleinberg, J. & Lan, X. (2006). Group formation in large social networks: membership, growth, and evolution. Em *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 44--54. ACM.

Barabäsi, A.; Jeong, H.; Néda, Z.; Ravasz, E.; Schubert, A. & Vicsek, T. (2002). Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3-4):590--614.

Barton, J. & Vijayaraghavan, V. (2002). Ubiwise, a ubiquitous wireless infrastructure simulation environment. *HP Labs*.

Battiti, R. & Bertossi, A. (1999). Greedy, prohibition, and reactive heuristics for graph partitioning. *Computers, IEEE Transactions on*, 48(4):361--385.

Baumgart, I.; Heep, B. & Krause, S. (2007). Oversim: A flexible overlay network simulation framework. Em *IEEE Global Internet Symposium, 2007*, pp. 79--84. IEEE.

Berners-Lee, T. & Mendelsohn, N. (2006). The rule of least power.

Bottazzi, D.; Corradi, A. & Montanari, R. (2006). Context-aware middleware solutions for anytime and anywhere emergency assistance to elderly people. *Communications Magazine, IEEE*, 44(4):82--90.

Bouzy, B. & Cazenave, T. (1997). Using the object oriented paradigm to model context in computer go. Em *Proceedings of the First International and Interdisciplinary Conference on Modeling and Using Context*.

Boyd, D. & Heer, J. (2006). Profiles as conversation: Networked identity performance on friendster. Em *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 3, pp. 59c--59c. IEEE.

Buchegger, S. & Datta, A. (2009). A case for p2p infrastructure for social networks-opportunities & challenges. Em *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, pp. 161--168. IEEE.

Bui, T. & Moon, B. (1996). Genetic algorithm and graph partitioning. *Computers, IEEE Transactions on*, 45(7):841--855.

Capra, L.; Emmerich, W. & Mascolo, C. (2001). Reflective middleware solutions for context-aware applications. *Metalevel Architectures and Separation of Crosscutting Concerns*, pp. 126--133.

Chaintreau, A.; Fraigniaud, P. & Lebhar, E. (2008). Opportunistic spatial gossip over mobile social networks. Em *Proceedings of the first workshop on Online social networks*, pp. 73--78. ACM.

Chang, W. (2008). Catpro: context-aware task prompting system with multiple modalities for individuals with cognitive impairments. Em *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, pp. 301--302. ACM.

Chen, G. & Kotz, D. (2000). A survey of context-aware mobile computing research.

Chen, H. (2005). Using owl in a pervasive computing broker. Relatório técnico, DTIC Document.

Cheng, X.; Dale, C. & Liu, J. (2008). Statistics and social network of youtube videos. Em *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pp. 229--238. Ieee.

Christ, A. (2011). Bridging the mobile app gap. *Connectivity and the User Experience*, p. 27.

Christakis, N. & Fowler, J. (2007). The spread of obesity in a large social network over 32 years. *New England Journal of Medicine*, 357(4):370--379.

Clarke, I.; Sandberg, O.; Wiley, B. & Hong, T. (2001). Freenet: A distributed anonymous information storage and retrieval system. Em *Designing Privacy Enhancing Technologies*, pp. 46--66. Springer.

Clocksin, W. & Mellish, C. (2003). *Programming in PROLOG*. Springer.

Costa, R. A. (2012). *Uma análise do uso de redes sociais como ferramenta Uma análise do uso de redes sociais como ferramenta para gestão do conhecimento*. Tese de doutorado, Universidade Federal de Pernambuco.

Davies, N. & Gellersen, H. (2002). Beyond prototypes: Challenges in deploying ubiquitous systems. *Pervasive Computing, IEEE*, 1(1):26--35.

Davis, G.; Yoo, M. & Baker, W. (2003). The small world of the american corporate elite, 1982-2001. *Strategic organization*, 1(3):301--326.

Davis, M.; Good, N. & Sarvas, R. (2004). From context to content: Leveraging context for mobile media metadata. Em *International Multimedia Conference: 12th annual ACM international conference on Multimedia*.

Dey, A. (2001). Understanding and using context. *Personal and ubiquitous computing*, 5(1):4--7.

Dhillon, I. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. Em *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 269--274. ACM.

Ding, C.; He, X.; Zha, H.; Gu, M. & Simon, H. (2001). A min-max cut algorithm for graph partitioning and data clustering. Em *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pp. 107--114. IEEE.

Droms, R. (1993). Dynamic host configuration protocol.

Erdös, P. & Rényi, A. (1959). On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290--297.

Firth, R. (2004). *Elements of social organization*, volume 2. Routledge.

Ford Jr, L. & Fulkerson, D. (1958). Constructing maximal dynamic flows from static flows. *Operations Research*, pp. 419--433.

Franklin, S. & Graesser, A. (1997). Is it an agent, or just a program?: A taxonomy for autonomous agents. *Intelligent Agents III Agent Theories, Architectures, and Languages*, pp. 21--35.

Gamma, E.; Helm, R.; Johnson, R. & Vlissides, J. (1994). Gang of four. *Design Patterns: Elements of Reusable Object-Oriented Software*.

Garcia-Luna-Aceves, J. & Madruga, E. (1999). The core-assisted mesh protocol. *Selected Areas in Communications, IEEE Journal on*, 17(8):1380--1394.

Garey, M.; Johnson, D. & Stockmeyer, L. (1976). Some simplified np-complete graph problems. *Theoretical computer science*, 1(3):237--267.

Ghiasi, S.; Srivastava, A.; Yang, X. & Sarrafzadeh, M. (2002). Optimal energy aware clustering in sensor networks. *Sensors*, 2(7):258--269.

Ghidini, C. & Giunchiglia, F. (2001). Local models semantics, or contextual reasoning= locality+ compatibility. *Artificial intelligence*, 127(2):221--259.

Ghinita, G.; Kalnis, P. & Skiadopoulos, S. (2007). Mobihide: A mobilea peer-to-peer system for anonymous location-based queries. *Advances in Spatial and Temporal Databases*, pp. 221--238.

González, M.; Lind, P. & Herrmann, H. (2006). System of mobile agents to model social networks. *Physical review letters*, 96(8):88702.

Gray, P. & Salber, D. (2001). Modelling and using sensed context information in the design of interactive applications. *Engineering for Human-Computer Interaction*, pp. 317--335.

Greenhalgh, A. (2007). icompete: Analyzing vendor-exclusive smartphone tying arrangements under federal law. *Loy. Consumer L. Rev.*, 20:438.

Gu, T.; Tan, E.; Pung, H. & Zhang, D. (2005). A peer-to-peer architecture for context lookup. Em *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pp. 333--341. IEEE.

Guo, C.; Zhong, L. & Rabaey, J. (2001). Low power distributed mac for ad hoc sensor radio networks. Em *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE*, volume 5, pp. 2944--2948. IEEE.

Gupta, A.; Kalra, A.; Boston, D. & Borcea, C. (2009). Mobisoc: a middleware for mobile social computing applications. *Mobile Networks and Applications*, 14(1):35--52.

Harvey, W. (1995). *Nonsystematic backtracking search*. Tese de doutorado, stanford university.

Hempel, J. & Lehman, P. (2005). The myspace generation. *Business Week*, 3963:88--93.

Hitzler, P.; Krötzsch, M.; Parsia, B.; Patel-Schneider, P. F. & Rudolph, S. (2009). Owl 2 web ontology language primer. *W3C Working Draft*, 21.

Horrocks, I.; Parsia, B.; Patel-Schneider, P. & Hendler, J. (2005). Semantic web architecture: Stack or two towers? *Principles and Practice of Semantic Web Reasoning*, pp. 37--41.

Ioannidis, S.; Chaintreau, A. & Massoulié, L. (2009). Optimal and scalable distribution of content updates over a mobile social network. Em *INFOCOM 2009, IEEE*, pp. 1422--1430. IEEE.

Jackson, N. (2011). Infographic: App store growth compared to mcdonald's burgers - nicholas jackson - technology - the atlantic.

Jansen, N. (2011). Design and implementation of a web gateway for mobile collaboration services.

Kagal, L.; Korolev, V.; Chen, H.; Joshi, A. & Finin, T. (2001). Centaurus: A framework for intelligent services in a mobile environment. Em *Distributed Computing Systems Workshop, 2001 International Conference on*, pp. 195--201. IEEE.

Katz, J. (1994). Geographical proximity and scientific collaboration. *Scientometrics*, 31(1):31--43.

Kern, N. & Schiele, B. (2003). Context-aware notification for wearable computing. Em *Proceedings of the 7th IEEE International Symposium on Wearable Computers (ISWC'03)*, pp. 223--230.

Kern, N.; Schiele, B. & Schmidt, A. (2003). Multi-sensor activity context detection for wearable computing. *Ambient Intelligence*, pp. 220--232.

Kernighan, B. & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291--307.

Khaddage, F. & Lattemann, C. (2009). Towards an ad-hoc mobile social learning network using mobile phones. *INTERACTIVE COMPUTER AIDED LEARNING*, 12:374--380.

Kidd, C.; Orr, R.; Abowd, G.; Atkeson, C.; Essa, I.; MacIntyre, B.; Mynatt, E.; Starner, T. & Newstetter, W. (1999). The aware home: A living laboratory for ubiquitous computing research. *Cooperative buildings. Integrating information, organizations, and architecture*, pp. 191--198.

Kirkpatrick, D. (2011). *The Facebook effect: The inside story of the company that is connecting the world*. Simon and Schuster.

Kochen, M. (1989). *The small world*. Ablex Publishing.

Kofod-Petersen, A. & Mikalsen, M. (2005). Context: Representation and reasoning–representing and reasoning about context in a mobile environment. Em *Revue d'Intelligence Artificielle*. Citeseer.

Kotz, D.; Henderson, T.; Abyzov, I. & Yeo, J. (2009). CRAWDAD data set dartmouth/campus (v. 2009-09-09). Downloaded from http://crawdad.cs.dartmouth.edu/dartmouth/campus.

Krause, A.; Smailagic, A. & Siewiorek, D. (2006). Context-aware mobile computing: Learning context-dependent personal preferences from a wearable sensor array. *Mobile Computing, IEEE Transactions on*, 5(2):113--127.

Kraut, R.; Egido, C. & Galegher, J. (1988). Patterns of contact and communication in scientific research collaboration. Em *Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, pp. 1--12. ACM.

Kwak, H.; Lee, C.; Park, H. & Moon, S. (2010). What is twitter, a social network or a news media? Em *Proceedings of the 19th international conference on World wide web*, pp. 591--600. ACM.

Kwiatkowska, M.; Norman, G. & Parker, D. (2002). Prism: Probabilistic symbolic model checker. *Computer Performance Evaluation: Modelling Techniques and Tools*, pp. 113--140.

Leskovec, J.; Backstrom, L.; Kumar, R. & Tomkins, A. (2008). Microscopic evolution of social networks. Em *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 462--470. ACM.

Leskovec, J.; Kleinberg, J. & Faloutsos, C. (2007). Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2.

Levine, S. & Kurzban, R. (2006). Explaining clustering in social networks: Towards an evolutionary theory of cascading benefits. *Managerial and Decision Economics*, 27(2-3):173--187.

Liben-Nowell, D.; Novak, J.; Kumar, R.; Raghavan, P. & Tomkins, A. (2005). Geographic routing in social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11623.

Lien, Y.; Jang, H. & Tsai, T. (2009). A manet based emergency communication and information system for catastrophic natural disasters. Em *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on*, pp. 412--417. IEEE.

Liljeros, F.; Edling, C.; Amaral, L.; Stanley, H. & Aberg, Y. (2001). The web of human sexual contacts. *Arxiv preprint cond-mat/0106507*.

Lubke, R.; Schuster, D. & Schill, A. (2011). Mobilisgroups: Location-based group formation in mobile social networks. Em *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pp. 502--507. IEEE.

Mani, M.; Ngyuen, A. & Crespi, N. (2009). What's up: P2p spontaneous social networking. Em *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pp. 1--2. Ieee.

McCanne, S.; Floyd, S.; Fall, K.; Varadhan, K. et al. (1997). Network simulator ns-2.

McCarthy, J. (1993). Notes on formalizing context.

McGuinness, D.; Van Harmelen, F. et al. (2004). Owl web ontology language overview. *W3C recommendation*, 10:2004--03.

Mokhtar, S.; McNamara, L. & Capra, L. (2009). A middleware service for pervasive social networking. Em *Proceedings of the International Workshop on Middleware for Pervasive Mobile and Embedded Computing*, p. 2. ACM.

Montresor, A. & Jelasity, M. (2009). Peersim: A scalable p2p simulator. Em *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*, pp. 99--100. Ieee.

Motik, B.; Patel-Schneider, P.; Parsia, B.; Bock, C.; Fokoue, A.; Haase, P.; Hoekstra, R.; Horrocks, I.; Ruttenberg, A.; Sattler, U. et al. (2009). Owl 2 web ontology language: Structural specification and functional-style syntax. *W3C Recommendation*, 27.

Mtibaa, A.; Chaintreau, A.; LeBrun, J.; Oliver, E.; Pietilainen, A. & Diot, C. (2008). Are you moved by your social network application? Em *Proceedings of the first workshop on Online social networks*, pp. 67--72. ACM.

Nardi, B. (1996). *Context and consciousness: activity theory and human-computer interaction*. The MIT Press.

Newman, M. (2001). The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404.

Newman, M. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577--8582.

Ng, A.; Jordan, M. & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849--856.

Nolan, T. (2011). A smarter way to practise. *BMJ*, 342.

Oehlman, D. & Blanc, S. (2011). Native bridging with phonegap. *Pro Android Web Apps*, pp. 193--219.

Pancake, C. & Utter, S. (1991). A bibliography of parallel debuggers. *ACM SIGPLAN Notices*, 26(1):21--37.

Perkins, C. (1998). Mobile networking through mobile ip. *Internet Computing, IEEE*, 2(1):58--69.

Pietiläinen, A.; Oliver, E.; LeBrun, J.; Varghese, G. & Diot, C. (2009). Mobiclique: middleware for mobile social networking. Em *Proceedings of the 2nd ACM workshop on Online social networks*, pp. 49--54. ACM.

Piorkowski, M.; Raya, M.; Lugo, A.; Papadimitratos, P.; Grossglauser, M. & Hubaux, J. (2008). Trans: realistic joint traffic and network simulator for vanets. *ACM SIGMOBILE Mobile Computing and Communications Review*, 12(1):31--33.

Pongor, G. (1993). Omnet: Objective modular network testbed. Em *Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and*

Telecommunication Systems, pp. 323--326. Society for Computer Simulation International.

Postel, J. (1980). User datagram protocol. *Isi*.

Pradhan, S. (2000). Semantic location. *Personal and Ubiquitous Computing*, 4(4):213--216.

Preis, R. & Diekmann, R. (1997). Party-a software library for graph partitioning. Em *Advances in Computational Mechanics with Parallel and Distributed Processing*. Citeseer.

Preuveneers, D.; Van den Bergh, J.; Wagelaar, D.; Georges, A.; Rigole, P.; Clerckx, T.; Berbers, Y.; Coninx, K.; Jonckers, V. & De Bosschere, K. (2004). Towards an extensible context ontology for ambient intelligence. *Ambient intelligence*, pp. 148--159.

Raacke, J. & Bonds-Raacke, J. (2008). Myspace and facebook: Applying the uses and gratifications theory to exploring friend-networking sites. *CyberPsychology & Behavior*, 11(2):169--174.

Ratnasamy, S.; Francis, P.; Handley, M.; Karp, R. & Shenker, S. (2001). A scalable content-addressable network. *ACM SIGCOMM Computer Communication Review*, 31(4):161--172.

Ripeanu, M. (2001). Peer-to-peer architecture case study: Gnutella network. Em *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pp. 99--100. IEEE.

Rowstron, A. & Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. Em *Middleware 2001*, pp. 329--350. Springer.

Ryan, N. (1999). Contextml: Exchanging contextual information between a mobile client and the fieldnote server. *1999*.

Samulowitz, M.; Michahelles, F. & Linnhoff-Popien, C. (2002). Capeus: An architecture for context-aware selection and execution of services. *New developments in distributed applications and interoperable systems*, pp. 23--39.

Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4):10--17.

Schilit, B.; Adams, N. & Want, R. (1994). Context-aware computing applications. Em *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pp. 85--90. IEEE.

Schilit, W. (1995). *A system architecture for context-aware mobile computing*. Tese de doutorado, Columbia University.

Schmidt, A.; Beigl, M. & Gellersen, H. (1999). There is more to context than location. *Computers & Graphics*, 23(6):893--901.

Schuster, D.; Springer, T. & Schill, A. (2010). Service-based development of mobile real-time collaboration applications for social networks. Em *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pp. 232--237. IEEE.

Scowen, R. (1995). Prolog—part 1, general core. *ISO/IEC*, pp. 13211--1.

Shah, R. & Rabaey, J. (2002). Energy aware routing for low energy ad hoc sensor networks. Em *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 1, pp. 350--355. Ieee.

Shaw, M. & Garlan, D. (1996). *Software architecture: perspectives on an emerging discipline*, volume 123. Prentice Hall.

Shen, H.; Shu, Y. & Yu, B. (2004). Efficient semantic-based content search in p2p network. *Knowledge and Data Engineering, IEEE Transactions on*, 16(7):813--826.

Solomon, J. (1997). *Mobile IP: the Internet unplugged*. Prentice-Hall, Inc.

Starner, T. (1996). Human-powered wearable computing. *IBM systems Journal*, 35(3.4):618--629.

Starner, T.; Mann, S.; Rhodes, B.; Levine, J.; Healey, J.; Kirsch, D.; Picard, R. & Pentland, A. (1997). Augmented reality through wearable computing. *Presence: Teleoperators and Virtual Environments*, 6(4):386--398.

Steinbach, M.; Karypis, G.; Kumar, V. et al. (2000). A comparison of document clustering techniques. Em *KDD workshop on text mining*, volume 400, pp. 525--526. Boston.

Stoica, I.; Morris, R.; Karger, D.; Kaashoek, M. & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149--160.

Strang, T. & Linnhoff-Popien, C. (2003). Service interoperability on context level in ubiquitous computing environments. Em *Intl. Conf. on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003w)*.

Strang, T. & Linnhoff-Popien, C. (2004). A context modeling survey. Em *Workshop on Advanced Context Modelling Reasoning and Management as part of UbiComp*, pp. 1--8. Citeseer.

Strang, T.; Linnhoff-Popien, C. & Frank, K. (2003). Cool: A context ontology language to enable contextual interoperability. Em *Distributed applications and interoperable systems*, pp. 236--247. Springer.

Tapia, E.; Intille, S. & Larson, K. (2004). Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing*, pp. 158--175.

Tärnlund, S. (1977). Horn clause computability. *BIT Numerical Mathematics*, 17(2):215--226.

Thijsse, B. (2002). Spline2 v6. 0 tutorial and user manual–unix version. *Delft University of Technology (Feb. 1, 2002) http://www. 3me. tudelft. nl/live/pagina. jsp*.

Toninelli, A.; Pathak, A.; Seyedi, A.; Cardoso, R. & Issarny, V. (2010). Middleware support for mobile social ecosystems. Em *Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual*, pp. 293--298. IEEE.

Triantafillou, P. & Aekaterinidis, I. (2004). Content-based publish-subscribe over structured p2p networks. Em *Proceedings of the third international workshop on distributed event-based systems (DEBS)*, pp. 104--109. Citeseer.

Tsai, F.; Han, W.; Xu, J. & Chua, H. (2009). Design and development of a mobile peer-to-peer social networking application. *Expert systems with applications*, 36(8):11077--11087.

Valkó, A. (1999). Cellular ip: A new approach to internet host mobility. *ACM SIGCOMM Computer Communication Review*, 29(1):50--65.

Vassiliadis, V.; Wielemaker, J. & Mungall, C. (2009). Processing owl2 ontologies using thea: An application of logic programming. Em *6th OWL Experiences and Directions Workshop (OWLED 2009)*.

Vaz de Melo, P. O. S. (2011). *On the Behavior of Rational Agents in Complex Networks*. Tese de doutorado, Universidade Federal de Minas Gerais.

Walsh, J.; Ahlberg, J. & Nilson, E. (1962). Best approximation properties of the spline fit. *J. Math. Mech*, 11(2):225--234.

Wang, B.; Bodily, J. & Gupta, S. (2004). Supporting persistent social groups in ubiquitous computing environments using context-aware ephemeral group service. Em *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pp. 287--296. IEEE.

Wang, S.; Chou, C.; Huang, C.; Hwang, C.; Yang, Z.; Chiou, C. & Lin, C. (2003). The design and implementation of the nctuns 1.0 network simulator. *Computer networks*, 42(2):175--197.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94--104.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75--84.

Wielemaker, J. (2003). An overview of the swi-prolog programming environment. Em *Proceedings of the 13th International Workshop on Logic Programming Environments*, pp. 1--16. Katholieke Universiteit Leuven, Heverlee, Belgium.

Zamir, O. & Etzioni, O. (1998). Web document clustering: A feasibility demonstration. Em *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 46--54. ACM.

Zhao, B.; Kubiatowicz, J.; Joseph, A. et al. (2001). Tapestry: An infrastructure for fault-tolerant wide-area location and routing.