André Ribeiro Coelho

COMUNICAÇÃO ORIENTADA A INTERESSE EM UM CONTEXTO DE COMPUTAÇÃO UBÍQUA E PERVASIVA

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Engenharia de Software.

Orientador: Prof. Dr. Orlando Gomes Loques Filho

Niterói

2013

Ficha Catalográfica – Esta página deve ser removida na versão a ser entregue para a banca, mas deve ser reinserida na versão final, com a ficha catalográfica fornecida pela biblioteca. Informações sobre este processo devem ser obtidas na secretaria da pós-graduação.

ANDRÉ RIBEIRO COELHO

COMUNICAÇÃO ORIENTADA A INTERESSE EM UM CONTEXTO DE COMPUTAÇÃO UBÍQUA E PERVASIVA

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Engenharia de Software.

Aprovada em dezembro de 2013.

BANCA EXAMINADORA

_____

Prof. Dr. Orlando Gomes Loques Filho – Orientador
UFF

_____

Prof. Dr. Claudio Luis de Amorim
COPPE/UFRJ

_____

Prof. Dr. Raphael Pereira de Oliveira Guerra
UFF

Niterói
2013

Ao meu amor Ana Carolina que nunca deixou de acreditar, de se preocupar e me incentivar e a minha família que sempre esteve ao meu lado me apoiando

AGRADECIMENTOS

A Deus, por não me deixar perder a fé e a esperança nos momentos mais difíceis.

Em especial ao professor Orlando, que acreditou em mim e me apoiou no momento que eu mais precisei no mestrado. Pela oportunidade que me foi dada de participar do grupo e de desenvolver este trabalho. Por sua disponibilidade em compartilhar o seu conhecimento e experiência com orientações que levarei pro resto da vida, tanto na vida pessoal quanto no ambiente profissional.

Em especial ao Héberte e ao Professor Cláudio Amorim da UFRJ, por todo o apoio e incansáveis ajudas em todas as minhas dúvidas. Sem vocês esse trabalho não seria possível.

Aos colegas do laboratório, que me deram toda a ajuda quando entrei no desenvolvimento deste projeto e que permitiram que eu ajudasse na construção e desenvolvimento do Framework.

Aos professores do Instituto de Computação da Universidade Federal Fluminense.

À Globo.com, que permitiu que eu investisse no meu desenvolvimento me liberando inúmeros dias ao longo desses anos.

Aos meus colegas de trabalho, que ao longo desses anos me incentivavam e motivavam para a finalização deste trabalho.

Em especial a minha namorada Ana Carolina, por sempre estar ao meu lado me incentivando, dando forças nos momentos mais difíceis e pela compreensão das noites, madrugadas e finais de semana que foram necessários nestes longos anos para a realização deste projeto.

À minha família, por ter me apoiado em todos os momentos.

La semplicità è la sofisticazione finale. (Leonardo da Vinci)

RESUMO

O desenvolvimento de aplicações ubíquas requer que os diversos dispositivos participantes obtenham dados do ambiente e se comuniquem, trocando e requisitando informações. Este trabalho compara o uso de duas implementações, conceitualmente equivalentes, do modelo de comunicação distribuída e orientada a interesses (também chamado de comunicação por eventos) aplicado em um ambiente ubíquo inteligente. Nesse modelo, a troca de mensagens ocorre de maneira indireta, de acordo com os interesses (ou eventos) definidos pelos participantes, ao invés de diretamente através da nomeação direta da origem e destino das entidades envolvidas na comunicação. Esse estilo de comunicação é conveniente pois permite o desacoplamento entre o emissor (*publisher*) e o subscritor (*subscriber*) do interesse, facilitando o atendimento de requisitos, tais como a difusão de informação entre participantes (potencialmente múltiplos) não previamente conhecidos. No trabalho inicialmente desenvolveu-se uma API de comunicação que fornece de forma transparente e unificada as mesmas funcionalidades para as duas implementações; em princípio, isso permite trocar a implementação utilizada através de uma simples mudança de parâmetros de configuração. Em seguida, para avaliar comparativamente as implementações, foram enunciadas questões de competência associadas a requisitos fundamentais de aplicações típicas em ambiente ubíquos. Com base nas questões de competência levantadas, foram separadamente desenvolvidas aplicações relevantes utilizando cada uma das implementações sob avaliação. O desenvolvimento dessas aplicações permitiu identificar vantagens e desvantagens na utilização de cada uma das propostas. Como resultado geral desse trabalho, mostra-se que no nível da comunicação as implementações são complementares. Diferenças advindas dos mecanismos de suporte (middleware e protocolos de comunicação) utilizados por cada implementação também foram identificadas.

Palavras-chave: computação ubíqua, ambientes inteligentes, smarthome, rede endereçada por interesses, SmartAndroid, REPI

ABSTRACT

The development of ubiquitous applications requires that the various participating devices obtain environment data and communicate with each other. This study compares the use of two implementations, conceptually equivalent, of the interest-oriented communication model (also know as communication by events), applied in an intelligent ubiquitous environment. In this model, message exchange occurs indirectly, according to the interests (or events) defined by participants, rather than directly through the direct appointment of the origin and destination of the entities involved in the communication. This approach is appropriate because it allows the decoupling between the publisher and the subscriber, facilitating compliance with requirements, such as dissemination of information among participants (potentially multiple) not previously known. At work was initially developed an API that provides communication in a transparent and unified way the same features for both implementations. In principle, this allows to change the implementation used through a simple change of configuration parameters. Next, to comparatively evaluate the implementations, competency questions were selected associated with the fundamental requirements of typical applications in ubiquitous environment. Based on the questions raised, were separately developed relevant applications using each of the implementations under evaluation. The development of applications allowed to identify advantages and disadvantages of using each of the proposals. As a result of this work, it is shown that in the level of communication implementations are complementary. Differences arising from the supporting mechanisms (middleware and communication protocols) used by each implementation have also been identified.

Keywords: ubiquitous computing, intelligent environments, smarthome, interest-oriented networks, SmartAndroid, REPI

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AmbI: Ambiente Inteligente;

API: Application Programming Interface;

AR: Agente de Recurso;

ARs: Agentes de Recurso;

DNS: Domain Name System;

IPGAP: Interface de Prototipagem de Gerenciamento de Aplicações Pervasivas;

JSON: JavaScript Object Notation;

RANS: Resource Agent Name System

REPI: Rede Endereçada por Interesses;

RPC: Remote Procedure Call;

SDR: Serviço de Descoberta de Recursos;

SLR: Serviço de Localização de Recursos;

SRR: Serviço de Registro de Recursos;

SCIADS: Sistema Computacional Inteligente de Assistência Domiciliar à Saúde

SUMÁRIO

CAPÍTULO 1 INTRODUÇÃO

1.1 MOTIVAÇÃO

O desenvolvimento de aplicações ubíquas proposto por Weiser [WEISER, M., 1991, WEISER, M., 1993, WEISER, M., 1994a, WEISER, M., 1994b] requer que os diversos dispositivos participantes obtenham dados do ambiente e se comuniquem. Weiser [WEISER, M., 1991] enfatiza que a tecnologia necessária para se obter um ambiente ubíquo baseia-se em fatores como preço, computação de baixo custo com telas convenientes, softwares para aplicações ubíquas e uma rede que ligue todos os dispositivos e softwares.

O surgimento de sistemas operacionais projetados para dispositivos móveis seguidos de um conjunto de interfaces de programação permitiram o crescimento de uma gama de aplicações que acompanham o usuário no seu dia a dia. A grande utilização destes dispositivos, assim como as ferramentas de desenvolvimento que eles fornecem, permitem um alcance maior e mais rápido de aplicações que enriquecem o ambiente do usuário da maneira mais transparente possível.

Além do desenvolvimento de aplicações que funcionem de forma mais transparente possível, Augusto [AUGUSTO, J. C.; MCCULLAGH, P., 2007] destaca a criação de sistemas que não necessitem que os usuários sejam especialistas em computação para beneficiarem-se do poder computacional. A área de Ambientes Inteligentes (AmbI) [AUGUSTO, J. C.; MCCULLAGH, P., 2007] visa enriquecer locais específicos tais como: quartos, casas, carros, prédios, ruas etc. com instalações computacionais que possam reagir às necessidades das pessoas e fornecer qualquer assistência necessária, minimizando as intervenções do usuário e utilizando essas ferramentas como mecanismo para a construção de aplicações.

Um exemplo proeminente de um AmbI é uma casa inteligente (ou *smarthome*). Segundo Park [PARK, J. et al., 2007], uma casa inteligente caracteriza-se por ser uma casa que inclui uma rede de dispositivos capazes de sensoriar os habitantes presentes e seu estado atual, fornecendo serviços apropriados a eles. As *smarthomes* vêm impulsionando a pesquisa na área de Ambientes Inteligentes, e outros projetos de pesquisa discorrem sobre o assunto [AUGUSTO, J. C.; MCCULLAGH, P., 2007].

O desenvolvimento, o uso e a validação de aplicações ubíquas representam um grande desafio tanto por parte dos desenvolvedores quanto por parte dos usuários. Para os desenvolvedores, é preciso ter acesso a recursos variados, dispositivos embarcados, espaço físico entre outros. Para os usuários, é necessário o acesso simples aos dispositivos, a capacidade de configurar e aplicar suas preferencias bem como a possibilidade de validação em um ambiente real de uso.

De modo a fornecer a infraestrutura necessária para criar aplicações sensíveis ao contexto com enfoque em Ambientes Inteligentes, [MARELI, D. et al., 2013, MARELI, D., 2013] propuseram um framework capaz de abstrair detalhes intrínsecos a esse tipo de aplicação, como obtenção de contexto, comunicação entre dispositivos, localização de recursos, entre outros, e cujo objetivo é permitir o desenvolvimento de aplicações que sigam os conceitos propostos por Weiser. Doravante, o framework desenvolvido por [MARELI, D. et al., 2013, MARELI, D., 2013] passa a ser referido também como Framework UFF.

No Framework UFF, um dos mecanismos de comunicação é por eventos ou interesses. Tal mecanismo segue um modelo de comunicação distribuída e orientada a interesses e é implementado através do paradigma *publish-subscribe*. Neste modelo, a comunicação não se dá de maneira tradicional origem-destino e sim, através de troca de mensagens que ocorrem de acordo com os interesses dos dispositivos da rede. Essa abordagem é importante pois permite o desacoplamento entre o emissor (*publish*) e o subscritor (*subscriber*), facilitando o atendimento de requisitos, tais como a difusão de informação entre participantes (potencialmente múltiplos) não previamente conhecidos.

Conceitualmente semelhante ao mecanismo de comunicação proposto por [MARELI, D. et al., 2013], [DUTRA, R. DE C. et al., 2012, DUTRA, R. DE C. et al., 2010] implementam um protocolo de comunicação capaz de estabelecer uma rede de comunicação também através de interesses. O Framework UFF e a proposta de [DUTRA, R. DE C. et al., 2012] diferem no fato do Framework UFF ser implementado por software em alto nível, enquanto a proposta de Dutra é implementada a nível de rede.

As características e o funcionamento do protocolo proposto por [DUTRA, R. DE C. et al., 2012] demonstram que o mesmo está aderente aos conceitos de comunicação orientada a interesses implementados no framework proposto por [MARELI, D. et al., 2013]. Desta forma, este protocolo poderia ser utilizado como

uma alternativa de implementação dos mecanismos de comunicação previstos pelo framework. Além disso, o suporte desenvolvido por [MORAES, H. F. et al., 2012a, MORAES, H. F. et al., 2012b] para a utilização do protocolo na internet possibilita e facilita a integração do protocolo com a implementação do framework.

1.2 OBJETIVO

Dado o exposto, este trabalho compara o uso de duas implementações conceitualmente equivalentes, do modelo de comunicação distribuída e orientada a interesses (também chamado de comunicação por eventos), aplicado em um ambiente ubíquo inteligente.

Para avaliar ambas as implementações, são levantadas questões de competência [GRÜNINGER, M. et al., 1995] que avaliam aspectos fundamentais de um ambiente ubíquo. De forma a validar as questões levantadas, foram desenvolvidas aplicações (veja Seção 6.2) utilizando cada uma das implementações avaliadas sob avaliação. O desenvolvimento dessas aplicações permitiu identificar vantagens e desvantagens na utilização de cada uma das propostas.

Além disso, uma avaliação quantitativa foi realizada visando identificar como o protocolo proposto por Dutra se comporta em um cenário típico de um ambiente ubíquo. Por fim, uma API de comunicação que fornece de forma transparente e unificada, as mesmas funcionalidades de ambas as implementações, foi desenvolvida.

1.3 ORGANIZAÇÃO

O restante deste trabalho está organizado em seis capítulos, além do capítulo de introdução. O Capítulo 2 apresenta uma visão geral da área de computação ubíqua e da área de ambientes inteligentes. Neste capítulo, é contextualizado o campo de pesquisa desse trabalho.

No Capítulo 3, a abordagem proposta por [MARELI, D. et al., 2013] é descrita. Todo o funcionamento do framework é apresentado, bem como a forma como a comunicação se dá no framework e na implementação do framework (SmartAndroid[1]).

No Capítulo 4, descreve-se a abordagem proposta por [DUTRA, R. DE C. et al., 2010]. Este capítulo apresenta os detalhes relevantes do funcionamento do

[1] http://www.tempo.uff.br/smartandroid/

protocolo REPI proposto, citando exemplos de como a comunicação é feita em um ambiente orientado a interesses. Além disso, é destacada a adaptação feita por [MORAES, H. F. et al., 2012a] para o funcionamento na internet atual.

No Capítulo 5, são apresentadas as alterações necessárias para se integrarem as abordagens propostas por [MARELI, D. et al., 2013] e [DUTRA, R. DE C. et al., 2010] [MORAES, H. F. et al., 2012a]. Neste capítulo, o mecanismo de comunicação desenvolvido para o correto funcionamento na SmartAndroid (implementação do framework) é exposto, assim como a API de integração com o protocolo REPI.

No Capítulo 6, é feita uma avaliação comparativa entre a abordagem utilizada por [MARELI, D. et al. , 2013] e a abordagem desenvolvida no Capítulo 5, em que toda a comunicação se dá através do protocolo REPI. São utilizadas diversas aplicações como base de avaliação. Estas aplicações foram escolhidas pois exploram questões de competência [GRÜNINGER, M. et al., 1995] típicas de um ambiente ubíquo e inteligente.

Finalmente, o Capítulo 7 conclui esta dissertação, destacando as contribuições deste trabalho, algumas limitações e sugerindo possíveis trabalhos futuros.

CAPÍTULO 2 CONCEITOS BÁSICOS

2.1 INTRODUÇÃO

A área de computação ubíqua (ou pervasiva) é embasada pelos trabalhos de Weiser na década de 90. Além dos trabalhos de Weiser, outros trabalhos [WELLNER, P. et al., 1993, DEY, A. et al., 2001, AUGUSTO, J. C.; MCCULLAGH, P., 2007], evidenciam o surgimento de uma área responsável por tornar o uso da tecnologia o mais transparente possível. Além disso, diversos trabalhos [WELLNER, P. et al., 1993, PARK, J. et al., 2007] argumentam a necessidade de se fornecer ambientes capazes de lidar com usuários potencialmente leigos. Esses ambientes específicos são enriquecidos de funcionalidades de forma que interação homem-máquina quase não se faça necessária, uma vez que o ambiente seria capaz de fornecer o que o usuário precisa sem uma interação explícita.

Neste Capítulo são apresentados os conceitos básicos utilizados ao longo deste trabalho. Na Seção 2.2 são apresentados os conceitos propostos por Weiser. Na Seção 2.3 é apresentada a área de ambientes inteligentes. Nas seções 2.4 e 2.5 é apresentado o conceito de contexto e evidenciado como ele é importante para o desenvolvimento de um ambiente inteligente. Na Seção 2.6 são apresentados desafios que justificam a criação de frameworks que auxiliam o desenvolvimento de aplicações ubíquas. Finalmente, na Seção 2.7, é destacado como a comunicação é importante neste ambiente.

2.2 COMPUTAÇÃO UBÍQUA

O desenvolvimento do ambiente ubíquo proposto por Weiser [WEISER, M., 1991, WEISER, M., 1993, WEISER, M., 1994a, WEISER, M., 1994b] requer que os diversos dispositivos participantes obtenham dados do ambiente e se comuniquem, trocando e requisitando informações de forma que a percepção de uso da tecnologia seja a mais transparente possível. A argumentação de Weiser [WEISER, M., 1991] baseia-se em uma consequência fundamental da psicologia em que, sempre que uma pessoa aprende algo suficientemente bem, ela perde a consciência da existência desse aprendizado, passando a ser um processo natural e transparente. Somente após este aprendizado, é possível focar em novas metas, fazendo uso do conhecimento adquirido sem se preocupar com ele.

Weiser já destaca em seu primeiro trabalho [WEISER, M., 1991] que a tecnologia necessária para se obter um ambiente ubíquo baseia-se em fatores como preço, computação de baixo custo com telas convenientes, softwares para aplicações ubíquas e uma rede que ligue todos os dispositivos e softwares.

Com os recentes e crescentes avanços dos dispositivos móveis, a tecnologia necessária vem tornando-se cada vez mais madura e acessível. Além disso, o surgimento de sistemas operacionais projetados para esses dispositivos, seguidos de um conjunto de interfaces de programação, permitiram o crescimento de uma gama de aplicações que acompanham o usuário no seu dia a dia.

Sistemas Operacionais como Google Android, Apple iOS e Microsoft Windows Phone vêm se popularizando nos últimos anos. Estes sistemas proveem inúmeras interfaces de programação de aplicativos que simplificam a integração com diversos sensores dos dispositivos, facilitando e impulsionando o desenvolvimento de novas aplicações cada vez mais ricas.

O uso adequado e direcionado desses dispositivos móveis pode e vem sendo usado como ferramenta para a criação de ambientes ubíquos e inteligentes. A grande utilização destes dispositivos, assim como as ferramentas de desenvolvimento que eles fornecem, permitem um alcance maior e mais rápido de aplicações que enriquecem o ambiente do usuário da maneira mais transparente possível. Em Wellner [WELLNER, P. et al., 1993], é destacada a importância de um direcionamento de construção de aplicações e ferramentas que enriqueçam e facilitem as atividades do nosso dia a dia. Como exemplos, citam-se a computação ubíqua e os trabalhos com realidade aumentada.

2.3 AMBIENTES INTELIGENTES

Além do desenvolvimento de aplicações que funcionem de forma mais transparente possível, Augusto [AUGUSTO, J. C.; MCCULLAGH, P., 2007] destaca a criação de sistemas que não necessitem que os usuários sejam especialistas em computação para beneficiarem-se do poder computacional. Interfaces gráficas intuitivas, assim como interfaces multimodais inteligentes, tais como manipulação de objetos, detecção de gestos e expressão facial compõem um conjunto de ferramentas para a construção de sistemas que proveem computação para os usuários de uma forma menos intrusiva.

A área de Ambientes Inteligentes (AmbI) [AUGUSTO, J. C.; MCCULLAGH, P., 2007] visa enriquecer locais específicos tais como: quartos, casas, carros, prédios, ruas etc. com instalações computacionais que possam reagir às necessidades das pessoas.

A ideia básica existente nos AmbI é enriquecer um ambiente com sensores e dispositivos conectados através de uma rede, em um sistema capaz de tomar decisões em benefício do usuário de acordo com informações capturadas em tempo real, de histórico ou através de preferências. Sob o aspecto da interação do usuário com o AmbI, Augusto [AUGUSTO, J. C.; MCCULLAGH, P., 2007] destaca que, embora se tenha como motivação reduzir as interações homem-máquina, uma gama de usuários podem precisar ou voluntariamente interagir com o sistema, indicando preferências ou necessidades específicas.

Um exemplo proeminente de um AmbI é uma casa inteligente (ou *smarthome*). Segundo Park [PARK, J. et al., 2007], uma casa inteligente caracteriza-se por ser uma casa composta de uma rede inteligente capaz de sensoriar os habitantes presentes e seu estado atual, fornecendo serviços apropriados a eles. As *smarthomes* vêm impulsionando a pesquisa na área de Ambientes Inteligentes, e diversos projetos de pesquisa discorrem sobre o assunto.

Apesar do grande enfoque que se confere às casas inteligentes, os requisitos, os conceitos e as soluções são muitas vezes aplicados a outros ambientes tais como: hospitais com sistemas de monitoramento, universidades com serviços tecnológicos, fábricas de produtos com sensores de acompanhamentos, transporte público com controle de fluxo, entre outros [AUGUSTO, J. C.; MCCULLAGH, P., 2007].

2.4 CONTEXTO

O desenvolvimento de ambientes inteligentes requer o conhecimento de informações que possam ser utilizadas na construção de aplicações inteligentes. Estruturando e sintetizando propostas anteriores, [DEY, A. et al., 2001] definem contexto como qualquer informação relevante que caracterize a situação de entidades. Sendo elas pessoas, lugares ou objetos tais quais sejam consideradas relevantes para a interação entre usuários e aplicações. No que tange o ambiente de uma casa inteligente, as pessoas são os indivíduos que interagem com o ambiente,

os lugares são os cômodos da casa, e os objetos são representações de objetos físicos associados a uma capacidade de comunicação e processamento.

O contexto é parte fundamental de um ambiente inteligente e pode ser obtido de diversas formas, porém tipicamente é obtido através de sensores instalados no ambiente ou de APIs de desenvolvimento que fornecem dados relativos aos objetos presentes no ambiente.

2.5 SENSIBILIDADE E INTERPRETAÇÃO AO CONTEXTO

O desenvolvimento de aplicações ubíquas e inteligentes depende não só da aquisição, como também da forma como as informações obtidas são utilizadas para o enriquecimento do ambiente. A sensibilidade ao contexto é a capacidade de adequar o funcionamento de um sistema de acordo com as informações de contexto obtidas a cada momento. Em [BALDAUF, M. et al., 2007], são expostas diversas propostas de frameworks que utilizam a sensibilidade ao contexto para o enriquecimento de um ambiente inteligente.

Pode-se citar como exemplo de sensibilidade e interpretação ao contexto em uma casa inteligente, o cenário onde uma pessoa tem o hábito de assistir séries de televisão com o ar-condicionado ligado ao chegar em casa. O último capítulo lançado da série de televisão já poderia estar disponível quando a pessoa chegasse em casa, bem como o ar-condicionado poderia ser ligado antes da chegada da pessoa (para gelar o ambiente previamente) ou no momento que a pessoa chegasse ao ambiente. Neste caso, é essencial que o hábito de assistir séries de televisão, o hábito de usar o ar-condicionado e a presença da pessoa, ou seja, o contexto, seja propagado pelo ambiente. Essa propagação e disponibilidade do contexto torna-se essencial para que aplicações inteligentes que possam, porventura, tomar decisões sem a intervenção do usuário tenham todas as informações necessárias, como no exemplo acima.

Destaca-se o trabalho de [RANGANATHAN, A. et al., 2003] o qual propõe a utilização de agentes sensíveis ao contexto capazes de interpretar e raciocinar sobre o contexto adquirido e assim tomar decisões pertinentes. Estes fornecem a base para a construção de sistemas mais complexos.

2.6 FRAMEWORKS PARA O DESENVOLVIMENTO E GERENCIAMENTO DE AMBIENTES INTELIGENTES

O desenvolvimento, o uso e a validação de aplicações ubíquas constituem um grande desafio tanto para os desenvolvedores quanto para os usuários. Aos desenvolvedores, é preciso ter acesso a recursos variados, dispositivos embarcados, espaço físico entre outros. Os usuários necessitam ter acesso simples aos dispositivos, ter a capacidade de configurar e aplicar suas preferências bem como a possibilidade de validação em um ambiente real de uso.

Nesse contexto, há dificuldades:

- de se estabelecer um protocolo comum de comunicação entre os componentes, dada a heterogeneidade dos dispositivos envolvidos;
- na interatividade das aplicações, pois depende fortemente da variedade de informações de contexto e serviços disponíveis no ambiente e
- no desenvolvimento e teste dessas aplicações, uma vez que dependem da disponibilidade de recursos como sensores (ex.: de iluminação, temperatura, presença), dispositivos embarcados e até mesmo espaços físicos tais como uma casa para aplicações do tipo *SmartHome*.

Visando abordar as questões acima, diversos frameworks de desenvolvimento e gerenciamento de ambientes inteligentes foram propostos [RANGANATHAN, A. et al., 2003, RANGANATHAN, A. et al., 2005, HELAL, S. et al., 2005, MARELI, D. et al., 2013]. Estes frameworks fornecem abstrações ou conceitos bem como mecanismos capazes de lidar com a diversidade dos dispositivos, a descentralização dos serviços, o suporte, a sensibilidade ao contexto e a prototipagem de aplicações ubíquas.

Em [RANGANATHAN, A. et al., 2003], são utilizadas ontologias para permitir um entendimento comum de diferentes contextos entre agentes distintos. As ontologias definem a estrutura e as propriedades de diferentes tipos de informação de contexto. O *middleware* desenvolvido foi integrado ao projeto Gaia, o qual permite enriquecer espaços inteligentes fornecendo a infraestrutura necessária. Este projeto usa CORBA para possibilitar a computação distribuída. Além disso, são utilizados agentes capazes de raciocinar sobre o contexto adquirido.

Em [RANGANATHAN, A. et al., 2005], é proposto um modelo de programação em alto nível, o qual endereça e facilita o processo de descoberta de entidades que satisfazem requisitos específicos. Além disso, são propostas operações de alto nível para a interação em um ambiente ubíquo.

Em [HELAL, S. et al., 2005], é proposta uma arquitetura genérica, aplicável a ambientes ubíquos. Os espaços inteligentes propostos funcionam tanto como um ambiente de execução, quanto uma biblioteca de desenvolvimento capaz de fornecer mecanismos de interação com o ambiente. Os serviços construídos pelos desenvolvedores podem ser encapsulados em aplicações que podem ser estendidas.

Em [MARELI, D. et al., 2013], é definido um framework conceitual capaz de endereçar questões relevantes para o desenvolvimento de aplicações ubíquas. Este destaca-se por tratar principalmente da heterogeneidade dos dispositivos, da variedade de informação de contexto e serviços, da disponibilidade de recursos e acessibilidade deles pelos usuários finais. Para isso, o framework adota uma arquitetura em camadas capaz de fornecer em conjunto uma abstração única. Assim como mecanismos de aquisição de contexto, capacidade de desenvolvimento de novos agentes e interfaces para localização e descoberta de recursos. Este framework foi utilizado como base para o desenvolvimento deste trabalho.

2.7 COMUNICAÇÃO EM AMBIENTES INTELIGENTES

A importância da aquisição de contexto destacada na Seção 2.4, visando proporcionar o enriquecimento do ambiente ubíquo através da sensibilidade e interpretação do contexto discutida na Seção 2.5, evidencia a importância de se fornecerem mecanismos de comunicação adequados ao desenvolvimento de aplicações ubíquas.

Além disso, questões pertinentes a um ambiente ubíquo, como a heterogeneidade dos dispositivos e seus respectivos protocolos de comunicação, estabelecem a relevância dos frameworks (discutidos na Seção 2.6) e de outras propostas na área de tratar especialmente a comunicação nestes ambientes. Na Seção 5.7 diversos trabalhos são abordados com enfoque em como a comunicação se realiza em suas respectivas propostas.

2.8 CONCLUSÃO DO CAPÍTULO

Este Capítulo apresentou os conceitos básicos utilizados neste trabalho. Caracterizou-se a área de Computação Ubíqua e de Ambientes Inteligentes destacando o ambiente de uma casa inteligente. Além disso, foi apresentado o conceito de contexto e a sensibilidade e interpretação de contexto em um possível cenário de atuação. Destacou-se a importância de frameworks que auxiliem o desenvolvimento de aplicações ubíquas bem como a relevância da comunicação nestes ambientes.

CAPÍTULO 3 FRAMEWORK DE DESENVOLVIMENTO DE APLICAÇÕES UBÍQUAS EM AMBIENTES INTELIGENTES

3.1 INTRODUÇÃO

No Capítulo anterior, foi demonstrada a importância de frameworks que suportem o desenvolvimento de aplicações para ambientes ubíquos e inteligentes. Estes frameworks endereçam questões comuns a estes ambientes. Em especial, o trabalho desenvolvido por [MARELI, D. et al., 2013] é capaz de tratar principalmente da heterogeneidade dos dispositivos, da variedade de informação de contexto e serviços e da disponibilidade de recursos e sua acessibilidade pelos usuários finais.

Neste Capítulo, é apresentado o Framework de Desenvolvimento de Aplicações Ubíquas em Ambientes Inteligentes proposto por [MARELI, D. et al., 2013]. Na Seção 3.2 é apresentado o que motivou a construção do framework. Na Seção 3.3 é apresentada a estrutura do framework. Na Seção 3.4 descreve-se a arquitetura utilizada no framework. Na Seção 3.5 são apresentados os dois mecanismos de comunicação que o framework oferece. Na Seção 3.6 é destacado o suporte a regras de contexto. Na Seção 3.7 é apresentada a interface de prototipagem desenvolvida, capaz de fornecer uma representação visual do ambiente, bem como fornecer uma ferramenta tanto para o desenvolvedor quanto para o usuário final. Na Seção 3.8 é apresentada a implementação do framework desenvolvida [MARELI, D. et al., 2013]. Finalmente na Seção 3.9 é detalhado o funcionamento da comunicação.

3.2 MOTIVAÇÃO DO FRAMEWORK

De modo a fornecer a infraestrutura necessária para criar aplicações sensíveis ao contexto com enfoque em Ambientes Inteligentes, [MARELI, D. et al., 2013] propuseram um framework capaz de abstrair detalhes intrínsecos a esse tipo de aplicação, como obtenção de contexto, comunicação entre dispositivos, localização de recursos, entre outros. Este framework [MARELI, D. et al., 2013] tem como objetivo permitir o desenvolvimento de aplicações que sigam os conceitos propostos por Weiser [WEISER, M., 1991, WEISER, M., 1993, WEISER, M., 1994a, WEISER, M., 1994b] aplicados a um contexto de Ambientes Inteligentes (AmbI) [AUGUSTO, J. C.; MCCULLAGH, P., 2007].

3.3 ESTRUTURA DO FRAMEWORK

Um dos grandes desafios de um AmbI é a heterogeneidade dos dispositivos presentes. De forma a padronizar os componentes de interação, [MARELI, D. et al., 2013] criaram o conceito de Agentes de Recurso (AR). Um Agente de Recurso é uma abstração capaz de representar um componente de um AmbI, assim como prover infraestrutura básica de interação com o framework proposto.

De maneira geral, um AR é composto de um nome único, uma hierarquia de tipos, variáveis de contexto e métodos ou serviços. Sendo assim, considerando-se um cenário de uma casa inteligente, por exemplo, que possui um AR "Fogão", que representa um fogão real, este poderia ter como nome "Fogão da Área de Churrasco", com variáveis de contexto que representem o estado das bocas do fogão, métodos para se ligar ou desligar o fogão, assim como poderia pertencer à hierarquia de tipos "\Eletrodomésticos\Alimentação\Fogão".

A hierarquia de tipos (associados a dispositivos típicos usados em ambientes inteligentes) proposta por [MARELI, D. et al., 2013] foi inspirada na ontologia descrita por [RAGANATHAM et al. 2005] e visa proporcionar uma classificação das entidades de modo que se possam construir consultas e instanciações de AR, tomando como base um determinado tipo.

3.4 ARQUITETURA DO FRAMEWORK

A arquitetura do framework é composta de três camadas principais: a camada inferior, representada pelos recursos presentes no AmbI, sendo eles possivelmente físicos, virtuais ou emulados; uma camada intermediária, composta por um middleware de suporte ao desenvolvimento e interação entre os componentes e uma terceira camada, composta pelas aplicações construídas, utilizando-se dos conceitos de computação ubíqua. Destaca-se que alguns agentes básicos já são oferecidos pelo framework, porém a implementação de novos agentes é plenamente suportada.

A estruturação em camadas utilizada no framework proposto por [MARELI, D. et al., 2013] foi concebida de modo a atender os requisitos relevantes para a modelagem e implementação de ambientes inteligentes. Especificamente, provê um middleware que inclui mecanismos para a construção de aplicações ubíquas, bem como suporte para interações com o ambiente, provendo também suporte a mecanismos de tomada de decisão que não dependam da interação explícita do usuário. Veja a Figura 1.

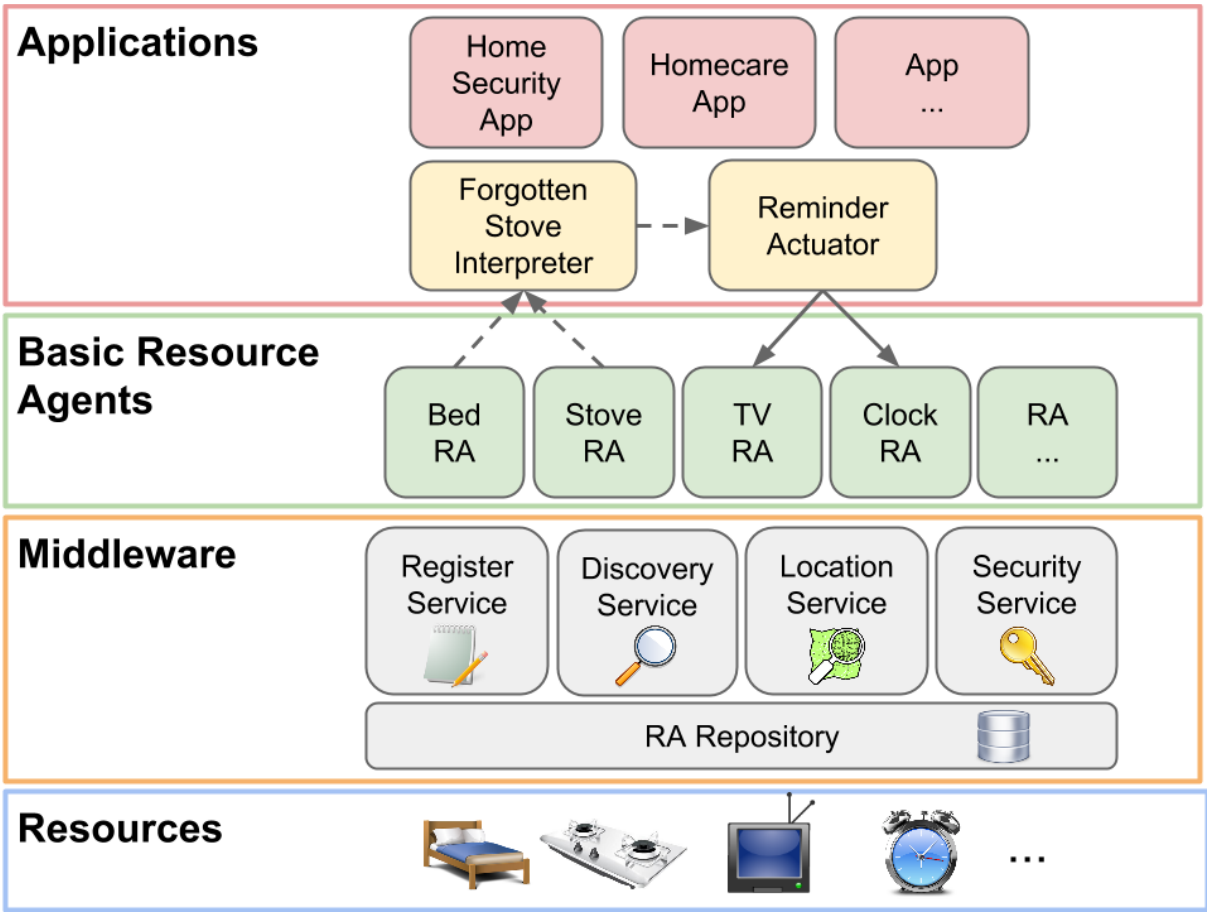

Figura 1 - Arquitetura do Framework [MARELI, D. et al., 2013]

Para controle dos agentes presentes no AmbI, um Agente de Recurso quando criado é Registrado em um Serviço de Registro de Recursos (SRR). Uma vez registrado, este AR pode ser consultado de duas formas: através de uma interface de consulta do Serviço de Descoberta de Recursos (SDR) ou através de uma interface de consulta do Serviço de Localização de Recursos (SLR). O SDR e o SLR proveem mecanismos de consulta de AR registrados no SRR.

O SDR é responsável pela localização de Recursos através de consultas por um identificador único ou pelo tipo do AR. Por outro lado, o SLR realiza consultas que identificam a localização física de um AR bem como consultas referentes aos AR mais próximos a uma determinada posição física no ambiente.

3.5 COMUNICAÇÃO NO FRAMEWORK

Um AR que deseja se comunicar com outro agente de recurso pode utilizar-se de dois mecanismos de comunicação. O primeiro é através de invocação remota de método (*Remote Procedure Call* ou RPC), e o segundo, através de comunicação por

eventos, utilizando o paradigma *publish-subscribe* [EUGSTER, P. T. et al. 2003]. Nesta Seção, o funcionamento dos dois mecanismos de comunicação fornecidos pelo framework é descrito.

3.5.1 INVOCAÇÃO REMOTA DE MÉTODO

A invocação remota de método é utilizada para executar uma ação síncrona no agente destino, sendo caracterizada por uma chamada a um método existente no agente destino. Esse mecanismo permite que as invocações locais feitas aos métodos sejam refletidas em invocações remotas de forma transparente. A classe que define o Agente de Recurso já possui toda a infraestrutura para suportar esse mecanismo.

Sendo assim, um Agente de Recurso de uma aplicação qualquer que representa, por exemplo, uma lâmpada está apto a invocar remotamente, de forma transparente, métodos providos pela lâmpada, tais como: “ligar” e “desligar”. O mesmo mecanismo poderia ser aplicado a um Agente de Recurso de uma aplicação qualquer que representa um Fogão na cozinha. Invocações a métodos locais como “acender uma boca do fogão” ou “apagar uma boca do fogão” serão automaticamente refletidos no Fogão.

Na Seção 3.9.1 é detalhado como a Invocação Remota de Método é implementada na instanciação do framework desenvolvida por [MARELI, D. et al., 2013].

3.5.2 COMUNICAÇÃO POR EVENTOS OU INTERESSES UTILIZANDO O PARADIGMA PUBLISH-SUBSCRIBE

A comunicação por eventos ou interesses é utilizada quando um AR deseja ser notificado de alterações em alguma variável de contexto do agente destino. Para isso, esse agente subscreve-se em uma variável de contexto, informando estar interessado nas alterações que ocorrem nesta variável. Quando ocorrer uma alteração na variável de contexto subscrita, os agentes interessados nesta variável serão imediatamente notificados. Veja na Figura 2 um exemplo do funcionamento da comunicação utilizando o paradigma *publish-subscribe*.

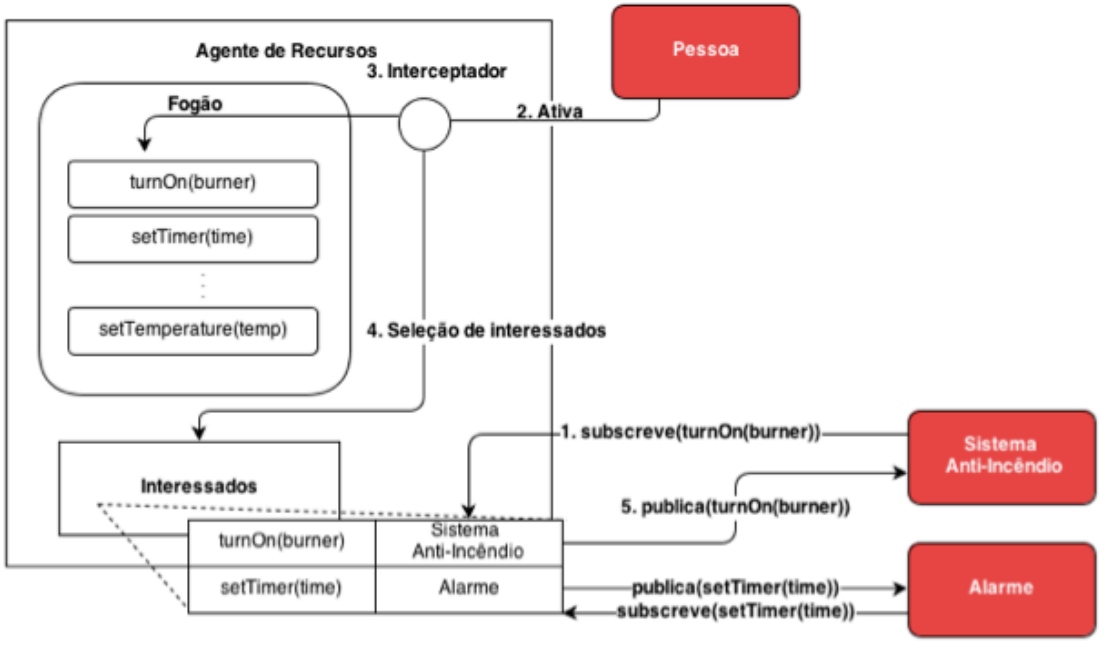

Figura 2 - Comunicação por eventos utilizando o paradigma *publish-subscribe*

A Figura 2 retrata um exemplo de aplicação de prevenção de incêndios que é capaz de monitorar dispositivos que são capazes de provocar um incêndio e, assim, acusar o risco através de um alarme. Neste cenário, o sistema de incêndio efetua uma subscrição na variável de contexto "*turnOn(burner)*" (Passo 1) de forma a ser notificado quando for efetuada uma alteração nesta variável. No momento em que uma pessoa liga o fogão (Passo 2), é identificado que ocorreu uma mudança nesta variável (Passo 3). Neste momento, todos os interessados nesta variável (Passo 4) serão notificados (Passo 5). O alarme só será notificado quando ocorrer uma mudança em "*setTimer(time)*".

Na Figura 2, é importante destacar a estrutura de dados dos agentes interessados que o AR armazena. Tal estrutura permite que assim que uma variável de contexto tenha uma alteração, a lista de interessados nela seja rapidamente identificada. Na figura, pode-se observar que há dois agentes interessados em variáveis de contexto distintas, o Sistema Anti-Incêndio interessado na variável "turnOn" e o Alarme interessado na variável "setTimer".

O sistema anti-incêndio poderia incluir outros sensores como fogo, fumaça, gases, entre outros e ter uma programação interna adequada para sua função de detectar incêndios. A subscrição de variáveis de contexto relacionadas aos sensores seria usada para receber as informações relevantes para a aplicação.

Na Seção 3.9.2 é detalhado como a Comunicação por eventos ou interesses utilizando o paradigma *publish-subscribe* é implementada na instanciação do framework desenvolvida por [MARELI, D. et al., 2013].

3.6 REGRAS DE CONTEXTO

A fim de correlacionar diversas informações de contexto, considerando critérios temporais, e extrair inteligência dessa correlação, [MARELI, D. et al., 2013] criaram uma infraestrutura que permite construir regras de contexto que se aplicam a determinadas condições lógicas e que são avaliadas por um Interpretador de contexto. Com isso, é possível modelarem-se regras que deixem o ambiente o mais ubíquo possível sem interação explícita do usuário.

Um exemplo de uma regra poderia ser “SE uma pessoa vai dormir E esquece o fogão ligado ENTAO dispara um alarme sonoro. Sendo assim, cenários como o exemplo citado na Seção 3.5 podem ser facilmente representados e ativados, desde que as informações de contexto estejam disponíveis no ambiente.

3.7 INTERFACE DE PROTOTIPAGEM

O desenvolvimento de aplicações ubíquas é muitas vezes prejudicado pela falta de ferramentas adequadas e de mecanismos de depuração [BARRETO, D. et al., 2013]. Além disso, a construção de um ambiente real para testes envolve, tipicamente, a aquisição de diversos dispositivos ou aparelhos que em muitas circunstâncias não são acessíveis a baixo custo, de maneira que representações em menor escala podem não refletir o ambiente real desejado [BARRETO, D. et al., 2013].

Tendo isso em mente, foi concebida uma Interface de Prototipagem e Gerenciamento de Aplicações Pervasivas (IPGAP) [BARRETO, D. et al., 2013] sobre a plataforma provida pelo framework que fornece uma representação visual das APIs e dos AR presentes em [MARELI, D. et al., 2013]. Nela é possível definir um mapa do ambiente, assim como os agentes nele presentes, interagindo diretamente com as representações visuais destes. Veja a Figura 3.

A IPGAP é capaz de criar tanto representações visuais de AR reais quanto de AR virtuais ou emulados. No caso de AR reais, qualquer operação na representação visual deste AR é refletida imediatamente no AR real através dos mecanismos de comunicação já descritos. O mesmo ocorre para alterações feitas no AR real; ou

seja, qualquer mudança ocorrida no agente real será refletida na IPGAP. Isto permite a visualização de um ambiente consistente, assim como fornece uma ferramenta importante tanto para projetistas de ambientes inteligentes ubíquos, quanto para desenvolvedores de aplicações ubíquas e até mesmo para usuários potencialmente leigos na área de computação.

Na Figura 3, é possível observar um mapa do ambiente inteligente, bem como os ARs presentes nele tais como: lâmpada, pessoa e tv. Na tela da IPGAP é possível ver as variáveis de contexto desses três ARs. Os três celulares apresentados estão atuando como simuladores de uma lâmpada, de uma tv e de um sensor de temperatura. Entretanto, é importante destacar que a IPGAP é capaz de lidar tanto com ARs reais quanto simulados.

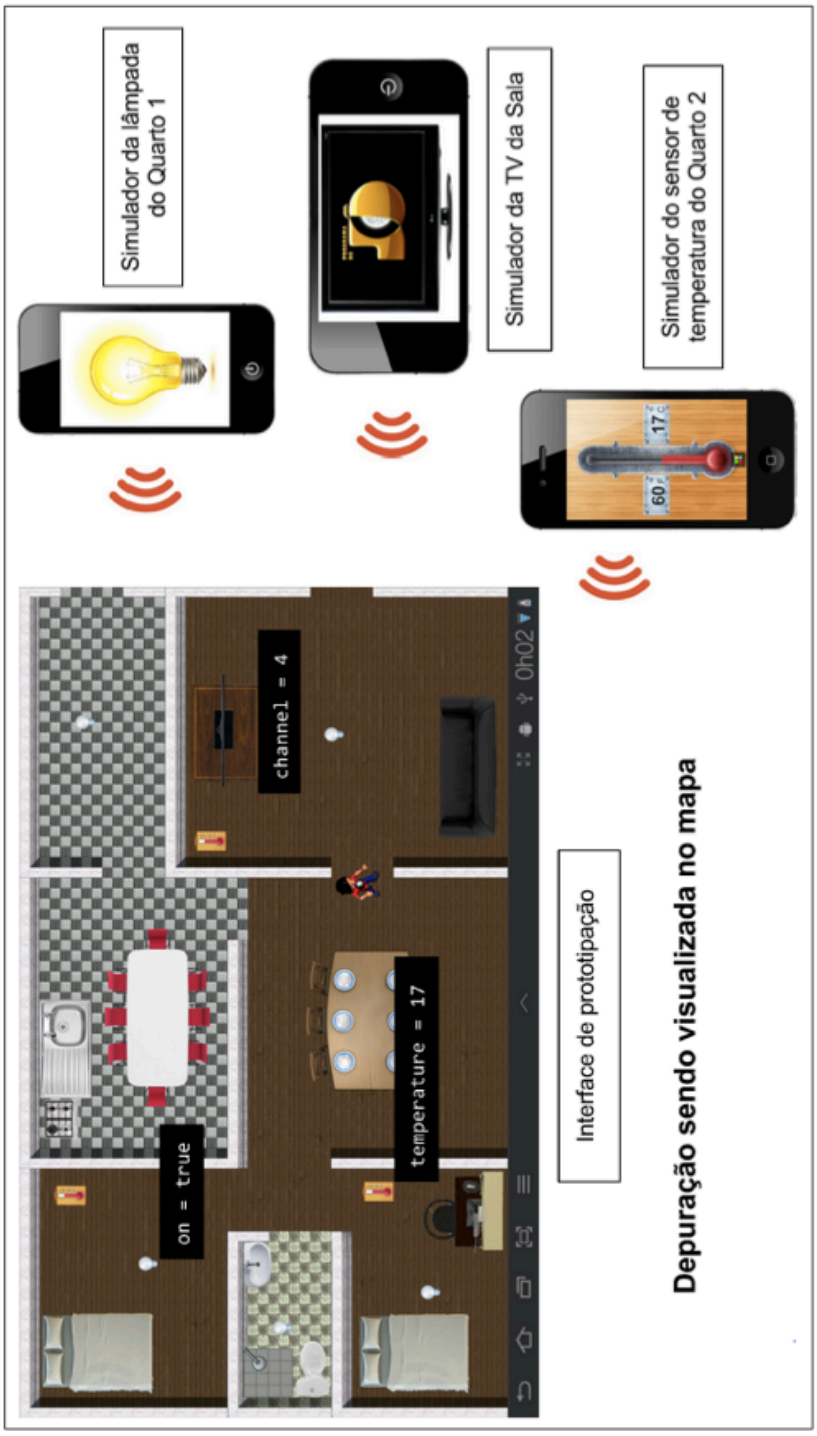

Figura 3 - Interface de Prototipagem [MARELI, D. et al., 2013]

3.8 IMPLEMENTAÇÃO DO FRAMEWORK

Visando consolidar e avaliar o framework proposto, [MARELI, D. et al., 2013] implementaram uma plataforma denominada SmartAndroid [MARELI, D. et al., 2013]. Esta plataforma é construída utilizando-se os conceitos de [MARELI, D. et al.,

2013] aplicados à infraestrutura proveniente da plataforma Android. Segundo [MARELI, D. et al., 2013], a escolha da plataforma Android teve como objetivo facilitar a construção rápida, assim como o beneficiamento da integração dos dispositivos disponíveis, bem como das opções de construção de sistemas embarcados. Destaca-se que [MARELI, D. et al., 2013] ressaltam que os conceitos propostos no framework são compatíveis com outras plataformas distribuídas atuais.

3.9 FUNCIONAMENTO DA COMUNICAÇÃO NA IMPLEMENTAÇÃO

Como visto na Seção 3.5, a comunicação no framework se dá através de dois mecanismos de comunicação: a invocação remota de método (veja Seção 3.5.1) e a comunicação por eventos utilizando o paradigma *publish-subscribe* [EUGSTER, P. T. et al. 2003] (veja Seção 3.5.2). Nesta Seção, é descrito como os mecanismos de comunicação foram implementados na SmartAndroid.

Para estabelecer um canal de comunicação comum entre os dispositivos presentes no ambiente inteligente, a SmartAndroid utiliza-se, por conveniência, de uma conexão a uma rede Wifi configurada com protocolo de segurança WPA. Em [MARELI, D. et al., 2013], são abordadas outras questões de segurança pertinentes a ambientes ubíquos e inteligentes. Sendo assim, no caso da SmartAndroid, toda comunicação se dá através da pilha de protocolos TCP/IP. É importante frisar que o framework não faz restrição quanto à tecnologia de rede utilizada.

3.9.1 INVOCAÇÃO REMOTA DE MÉTODO

Na Seção 3.5.1, é destacado que a invocação remota de método é um mecanismo de comunicação entre processos o qual permite que um procedimento seja executado remotamente, tipicamente em outro computador. Na SmartAndroid, é utilizado o protocolo Json-RPC por sua simplicidade, tamanho do pacote gerado e facilidade de uso.

Um *daemon* iniciado durante a inicialização do sistema é responsável por escutar mensagens Json-RPC em uma determinada porta e, assim, iniciar a execução de um comando local responsável por efetivar a chamada. A execução deste comando delega para um Dispatcher a responsabilidade de interpretar a chamada Json-RPC. O Dispatcher, por sua vez, desempacota a chamada Json-RPC e chama a execução do método invocado na mensagem. A resposta é devidamente empacotada no protocolo Json-RPC pelo Dispatcher e assim devolvida para quem

efetuou a chamada. A Figura 4 detalha as classes envolvidas neste fluxo de comunicação.

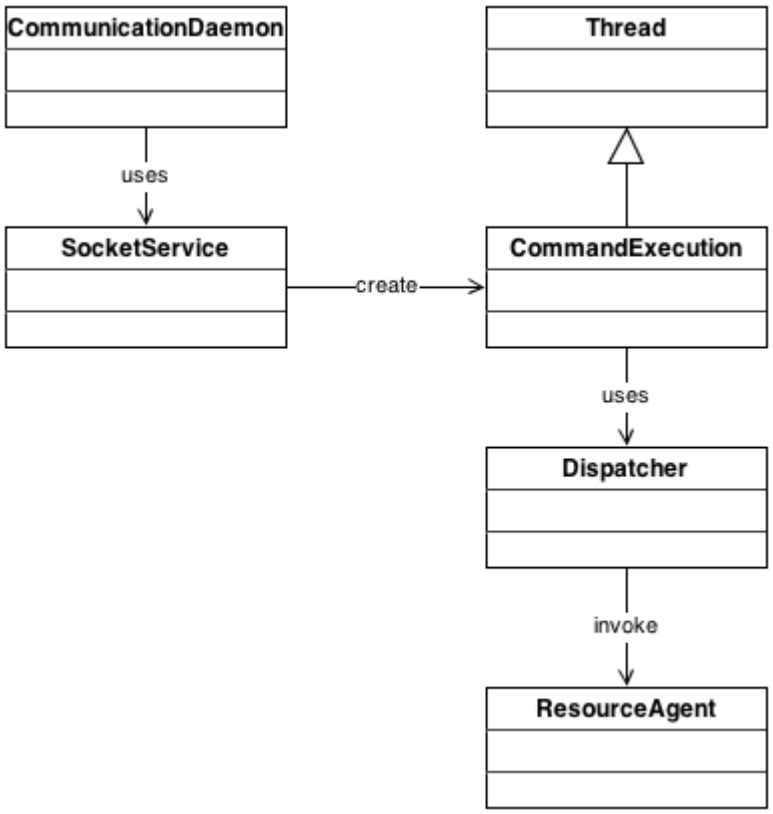

Figura 4 - Classes envolvidas no JsonRPC

Em termos de desenvolvimento, conforme descrito na Seção 3.5.1, as invocações remotas são feitas de forma transparente para o desenvolvedor que usa o middleware. Para isso, apenas é necessário criar um *Stub* do agente que se deseja controlar e herdá-lo de "ResourceAgentStub". Toda a infraestrutura de comunicação já estará disponível para uso. Qualquer chamada de método feita no *Stub* será diretamente e automaticamente refletida no Agente de Recurso controlado. Veja na Figura 5 um pseudocódigo demonstrando a utilização.

```
// instantiate the stub (proxy) from lamp
lamp = new LampStub(realLamp.getRans());

// register myAgent interest in "isOn" variable in the real lamp agent
lamp.registerStakeholder("isOn", myAgent.getRANS());

boolean on = lamp.isOn();
```

Figura 5 - Exemplo de uso do Stub

3.9.2 COMUNICAÇÃO POR EVENTOS OU INTERESSES UTILIZANDO O PARADIGMA PUBLISH-SUBSCRIBE

Na Seção 3.5.2, é destacado que a comunicação por eventos utilizando o paradigma *publish-subscribe* [Eugster, P. T. et al. 2003] é concebida através de um mecanismo onde um agente é capaz de registrar interesse em uma variável de contexto de outro agente. Na SmartAndroid, o agente que possui outros agentes interessados tem uma lista desses agentes para que, quando alterações sejam feitas nas suas respectivas variáveis de contexto, os devidos agentes sejam notificados através de mensagens Json-RPC.

Os Agentes de Recurso que, porventura, registraram interesses em variáveis de contexto de outros agentes implementam um método que é chamado, quando suas variáveis de contexto interessadas sofrerem alteração.

Em termos de desenvolvimento, cada Agente de Recurso e, consequentemente, cada *Stub* de Agente de Recurso possui um método "*registerStakeholder*". Para registrar interesse em uma variável de contexto de um agente, basta criar um *Stub* do AR interessado e chamar o método "*registerStakeholder*", informando a variável de contexto interessada. O AR que se registrou será notificado através do método "*notificationHandler*". Este receberá uma mensagem informando a variável de contexto que sofreu alteração e o valor dela. Veja na Figura 6 um pseudocódigo demonstrando a utilização.

Observa-se que as primitivas de registro e notificações podem ser usadas arbitrariamente nos códigos. Isso permite liberdade para o programador usá-las em aplicações fugindo ao padrão *publish-subscribe* proposto.

```
// instantiate the stub (proxy) from lamp
lamp = new LampStub(realLamp.getRans());

// register myAgent interest in "isOn" variable in the real lamp agent
lamp.registerStakeholder("isOn", myAgent.getRANS());

public class MyAgent extends ResourceAgent {
    ...

    @Override
    public void notificationHandler(String rai, String method, Object value) {
        if (method.equals("isOn")) {
            System.out.println(value)
        };
    }
}
```

Figura 6 - Exemplo de uso do notificationHandler

3.9.3 IDENTIFICAÇÃO DOS RECURSOS

Uma questão importante que afeta a comunicação é como os recursos de um sistema são identificados. Tipicamente a comunicação prevê que existam identificações únicas dos recursos. No framework esse problema é endereçado através de um conceito de Sistema de Nome de Agentes de Recursos (RANS) análogo ao Sistema de Nome de Domínios (DNS).

Quando um Agente de Recurso registra-se no Serviço de Registro de Recursos (SRR), ele é obrigado a informar um nome único em todo o ambiente da SmartAndroid. Como, para um AR ser identificável e comunicável, é necessário registrar-se no SRR, este torna-se um ponto único para registro e validação deste campo. O SRR possui o controle de todos os ARs cadastrados, assim como seus respectivos nomes, e o SDR possui as APIs de consulta adequadas.

Pode-se exemplificar o RANS através de nomes como: "lampadadasala.ra", "fogao.ra", "geladeira.ra", "lampadadoquarto.ra", entre outros. Destaca-se que os Serviços de Registro de Recursos (SRR), de Descoberta de Recursos (SDR) e de Localização de Recursos (SLR) possuem nomes fixos, não sendo possível nenhum outro agente cadastrar os mesmos nomes que eles.

Análogo ao DNS (Domain Name System) utilizado na internet, o RANS resolve o nome do agente, como por exemplo "lampadadasala.ra" para a informação onde o agente está localizado, que no caso da SmartAndroid é o IP do dispositivo. Também de forma análoga ao DNS, os dispositivos possuem um cache local contendo o registro dos RANS conhecidos. Caso um AR ainda não tenha se comunicado com um determinado RANS, o mecanismo de comunicação automaticamente efetua uma consulta no serviço de descoberta, pesquisando o registro desse RANS para assim estabelecer a comunicação, neste caso através do IP do dispositivo. Veja na Figura 7 um exemplo de envio de uma mensagem do agente “myAndroid.ra” para a “lampadadasala.ra” do ponto de vista do funcionamento do RANS.

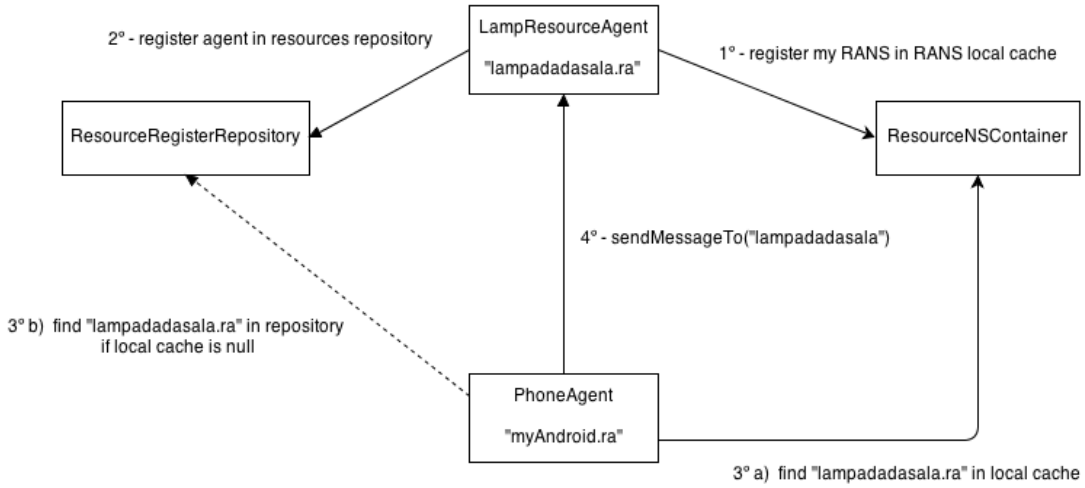

Figura 7 - RANS - Fluxo de resolução de nomes de agentes para envio de mensagens

Inicialmente um Agente de Recurso “lampadadasala.ra” recém criado registra o seu RANS em um cache local (Passo 1). Logo após, é feito o registro no repositório de recursos (Passo 2), para que qualquer outro AR saiba onde este RANS está localizado. Neste momento, qualquer agente pode estabelecer um canal de comunicação com o AR “lampadadasala.ra”. Se o AR “myAndroid.ra” desejar enviar uma mensagem para o AR “lampadadasala.ra”, o mesmo fará uma procura em seu cache local (Passo 3a) e, em caso de não encontrado, efetuará uma busca no serviço de descoberta por esse agente (Passo 3b). Tanto as consultas ao cache local quanto ao repositório de recursos retornam um registro do RANS, que neste caso é composto pelo nome do AR mais o IP onde o mesmo está localizado.

3.10 CONCLUSÃO DO CAPÍTULO

Este Capítulo apresentou o Framework de Desenvolvimento de Aplicações Ubíquas em Ambientes Inteligentes proposto por [MARELI, D. et al., 2013]. O conceito de Agentes de Recursos capazes de representar um componente de um Ambiente Inteligente representa a unidade básica de codificação e interação no framework. A arquitetura é composta em camadas, o que permite o desenvolvimento de novos componentes e aplicações utilizando uma infraestrutura básica. Além disso, é apresentada a implementação do framework (SmartAndroid) bem como o suporte dado a regras de contexto e a interface de prototipagem IPGAP desenvolvida.

Os mecanismos de comunicação do framework, seja através de invocação remota de método ou através de eventos ou interesses representam ferramentas importantes à troca de informações de contexto no ambiente. Tais mecanismos foram detalhados tanto no seu conceito como na forma como foram implementados na SmartAndroid. Destaca-se também como os Agentes de Recursos são identificados e como a sua identificação é utilizada no envio de mensagens entre eles.

CAPÍTULO 4 REPI - REDE ENDEREÇADA POR INTERESSES

4.1 INTRODUÇÃO

O Capítulo anterior detalha os mecanismos de comunicação adotados pelo framework de desenvolvimento de aplicações ubíquas em ambientes inteligentes. Através da comunicação por eventos, utilizando o paradigma *publish-subscribe,* é possível um Agente de Recurso registrar interesse em uma determinada variável de contexto e ser notificado de mudanças nela. O trabalho desenvolvido por [DUTRA, R. DE C. et al., 2010] desenvolve de forma semelhante, uma mecanismo de comunicação capaz de estabelecer uma rede de comunicação através de interesses. Esta proposta torna-se interessante, uma vez que é possível efetuar o endereçamento das mensagens enviadas através de interesses, não sendo necessário o conhecimento da origem-destino.

Neste Capítulo, é apresentado o Protocolo de Comunicação Orientado a Interesses proposto por [DUTRA, R. DE C. et al., 2010]. Na Seção 4.2 é apresentado a motivação para o estudo do protocolo. Nas seções 4.3 e 4.4, respectivamente são apresentados o protocolo e o seu funcionamento. Finalmente na Seção 4.5 é demonstrado como [MORAES, H. F. et al., 2012a] propôs um modelo P2P que cria uma rede sobreposta capaz de transpor o protocolo para a internet, bem como o protocolo funciona e se comporta neste cenário.

4.2 MOTIVAÇÃO

Em [MARELI, D. et al., 2013], é proposto um framework de desenvolvimento de aplicações ubíquas em ambientes inteligentes (veja Capítulo 3). Neste framework, um dos mecanismos de comunicação é por eventos ou interesses (veja Seção 3.4.2). Tal mecanismo é implementado através do paradigma publish-subscribe.

Conceitualmente semelhante ao mecanismo de comunicação proposto por [MARELI, D. et al., 2013], [DUTRA, R. DE C. et al., 2010] implementam um protocolo de comunicação capaz de estabelecer uma rede de comunicação também através de interesses.

Diante da semelhança conceitual, em alto nível, entre ambas as propostas, este trabalho visa avaliar a utilização do protocolo [DUTRA, R. DE C. et al., 2010]

como uma possível alternativa para os mecanismos de comunicação que [MARELI, D. et al., 2013] adotam. Para isso, este Capítulo, primeiramente, descreve o funcionamento e as características do protocolo proposto por [DUTRA, R. DE C. et al., 2010]. Além disso, a Seção 4.5 descreve como [MORAES, H. F. et al., 2012a] transpõem este protocolo para a estrutura atual da internet.

4.3 PROTOCOLO DE COMUNICAÇÃO ORIENTADO A INTERESSES

Inicialmente projetada para o compartilhamento de recursos [SOCOLOFSKY, T.; KALE, C., 1991][CALLON, R., 1990][CLARK, D., 1988] através de uma rede, a internet segue um modelo de comunicação resultante da comunicação entre duas máquinas [KOPONEN, T. et al., 2007][JACOBSON, V. et al., 2009]. Em outras palavras, uma máquina provê um serviço, e uma ou mais máquinas o acessam. Com o crescimento no uso de aplicações sociais como Youtube, Facebook, Instagram, Foursquare e de aplicações de busca de informação tal como Google, Yahoo e Bing, a internet vem desviando o foco de busca de uma máquina ou serviço para uma busca de informação ou conteúdo.

A Internet como conhecemos hoje não foi preparada para essa nova gama de aplicações, ou seja, a pilha de protocolos TCP/IP [SOCOLOFSKY, T.; KALE, C., 1991][CALLON, R., 1990] atualmente utilizada na internet não possui o suporte adequado para aplicações que explorem o estabelecimento de conexões entre usuários, através de vários tipos de relacionamentos [MORAES, H. F. et al., 2012a].

Em [DUTRA, R. DE C. et al., 2010], é desenvolvida uma proposta de um protocolo de comunicação que se diferencia dos demais por não utilizar o endereçamento origem-destino. O modelo proposto por [DUTRA, R. DE C. et al., 2010] estabelece que a comunicação seja endereçada através de interesses, ou seja, termos que representem os interesses dos usuários. Diante desse protocolo, é possível estabelecer conexões entre os usuários, a nível de rede, de interesses em comum, não sendo necessário o conhecimento da origem-destino.

Utilizando-se do mecanismo de interesses nas mensagens, [DUTRA, R. DE C. et al., 2010] baseia-se no modelo de *publish-subscribe* para permitir que os usuários troquem mensagens de acordo com os interesses assinados. Este modelo inicialmente desenvolvido com foco em redes ad hoc sem fio móveis (MANETs) não necessita conhecer a estrutura da rede para se enviar uma mensagem, uma vez que o mecanismo de encaminhamento se dá através dos interesses. A rede criada é

essencialmente colaborativa, ou seja, cada nó transmite suas mensagens a seus vizinhos, e esses vizinhos as propagam para seus vizinhos.

4.4 FUNCIONAMENTO DO PROTOCOLO

Na inicialização do sistema, cada dispositivo define um Prefixo Ativo (PA) composto de um Prefixo (P) mais um Interesse (I). O campo Prefixo (P) é composto de campos escolhidos probabilisticamente seguindo uma distribuição multivariada. Já o campo Interesse (I) é composto pelo interesse do usuário ou aplicação e é formado por qualquer sequência de caracteres. O campo P é utilizado tanto para o encaminhamento probabilístico das mensagens, definido pelo algoritmo do protocolo, quanto para identificar unicamente um nó através da sua sequência de bits. Como os campos do Prefixo (P) são escolhidos probabilisticamente, é possível (com uma probabilidade baixa, veja o Apêndice A3) que algumas mensagens não sejam entregues.

Utilizando-se de uma função de casamento, cada nó avalia se possui pelo menos um campo de seu prefixo P igual a um campo do prefixo P do nó de origem da mensagem. Caso ocorra o batimento, a mensagem é encaminhada para seus vizinhos, e, caso contrário, a mensagem é descartada. O repasse de uma mensagem é feito para a aplicação somente no caso de a aplicação estar interessada no campo I da mensagem recebida. Como o campo P é usado como identificação única, isto permite que o mesmo seja utilizado como meio de endereçamento em uma comunicação fim-a-fim permitindo, assim, dois nós se comunicarem diretamente.

A Figura 8 e a Tabela 1 baseadas no exemplo criado por [MORAES, H. F. et al., 2012a], exemplificam o funcionamento do protocolo. A linha tracejada define o alcance de cada dispositivo. A Tabela 1 representa o Prefixo Ativo (PA) de cada um dos dispositivos presentes na rede. Os campos Prefixos P1 e P2 são escolhidos probabilisticamente. É importante frisar que dois agentes podem ter partes do prefixo iguais.

Na Tabela 1, é possível constatar que os dispositivo A e D possuem o mesmo interesse. Contudo, por uma limitação física, ambos não possuem o alcance necessário para a transmissão da mensagem. Desta forma, é necessário que os demais membros da rede cooperem para que as mensagens transmitidas de A cheguem até D.

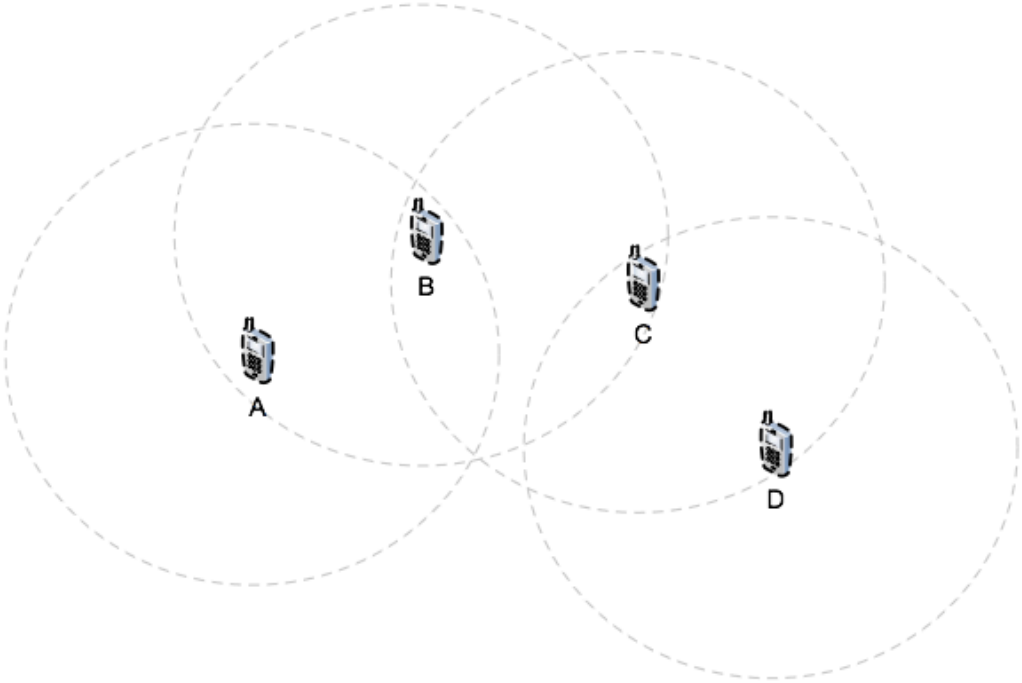

Figura 8 - Exemplo de uma rede REPI com 4 dispositivos [MORAES, H. F. et al., 2012a]

Dispositivo	Prefixo Ativo (PA)		
	Prefixo		Interesse
	P1	P2	
A	(4)	[3]	Vôlei
B	5	[3]	Carona
C	(4)	2	Jantar
D	5	1	Vôlei

Tabela 1 - Prefixos Ativos (PAs) dos 4 dispositivos

Ao enviar uma mensagem, o dispositivo A anexa ao seu cabeçalho o seu PA (4, 3, Vôlei). Por estar no seu raio de transmissão, o dispositivo B receberá essa mensagem. B, por sua vez, avalia os Prefixos e identifica que seu PA (5, 3, Carona) possui o mesmo prefixo "P2" (3) que A; com isso, B encaminha a mensagem de A para o dispositivo C. Da mesma maneira, C ao receber a mensagem de A, encaminhada pelo B, avalia o seu PA (4, 2, Jantar) e identifica que o prefixo "P1" (4) é comum ao prefixo de A, encaminhando, assim, a mensagem de A ao dispositivo D.

Como o interesse do dispositivo D (Vôlei) é o mesmo da mensagem (Vôlei) de A este nó aceita a mensagem e a repassa à aplicação. Como D não possui nenhum prefixo (5, 1) comum a mensagem de A (4, 3), a mensagem não é mais transmitida e, assim, descartada. Observa-se que os dispositivos B e C não possuíam um interesse na mensagem porém cooperam com a rede encaminhando-a.

4.5 REPI APLICADO A INTERNET

A fim de se transpor o protocolo criado por [DUTRA, R. DE C. et al., 2010] para a internet, [MORAES, H. F. et al., 2012a] propôs um modelo P2P que cria uma rede sobreposta elaborada com uma variação do algoritmo de formação de rede do Gnutella [MORAES, H. F. et al., 2012a]. Esta rede P2P é responsável por fornecer as características de funcionamento do protocolo criado na estrutura atual da internet. Essa adequação permite que o protocolo seja utilizado como uma alternativa viável na mudança de foco de comunicação origem-destino atrelada a dispositivos que se comunicam, para uma comunicação orientada a interesses definidos pelos usuários.

4.5.1 FORMAÇÃO DA REDE

Conforme destacado por [MORAES, H. F. et al., 2012a], a utilização de NAT (*Network Address Translator*) traz dificuldades bem conhecidas em uma comunicação P2P [HOLDREGE, M. et al., 2001][FORD, B. et al., 2005]. Para contornar essas dificuldades, [MORAES, H. F. et al., 2012a] utilizam a técnica de *hole punching* [FORD, B. et al., 2005], que prevê a utilização de um ponto de encontro globalmente conhecido para permitir a comunicação entre dois dispositivos. Desta forma, todo dispositivo que entrar na rede se comunica incialmente com um nó Origem de forma a iniciar o processo de descoberta de vizinhos. Por vizinhos entendem-se os dispositivos já presentes na rede.

O protocolo possui três grupos de mensagens: as mensagens de formação da rede, que possibilitam a criação de um canal de comunicação entre dois dispositivos; as mensagens de manutenção na rede, que permitem a manutenção dos canais de comunicação já criados e, por fim, as mensagens REPI que são as mensagens criadas pela aplicação.

A Figura 9 [MORAES, H. F. et al., 2012a] descreve o processo de descoberta de um vizinho. Nela as linhas tracejadas mais fortes representam as conexões já

estabelecidas, ou seja, A-Origem e B-Origem, enquanto as linhas tracejadas mais claras representam as novas conexões estabelecidas após o processo de descoberta de vizinhos. As demais linhas representam o processo de descoberta descrito a seguir.

A Figura 9 demonstra o processo de descoberta de um vizinho para um nó N que acabou de entrar na rede. Ele inicia o processo enviando uma mensagem ao nó Origem (Passo 1), que confirma o recebimento da mensagem (Passo 2). O nó N, por sua vez, solicita um novo vizinho ao nó Origem (Passo 3), que devolve o endereço do vizinho B (Passo 4). Destaca-se que no Passo 4 a escolha do vizinho visa proporcionar a melhor formação da rede. No mesmo momento, o nó Origem informa ao nó B o endereço do nó N (Passo 5) para que ambos possam se comunicar. Uma vez recebida pelos a mensagem do nó Origem pelos nós N e B, ambos enviam uma mensagem entre eles (Passos 6 e 8) para o estabelecimento de um novo canal de comunicação. Os nós N e B, por fim, confirmam o recebimento (Passos 7 e 9) e guardam em suas tabelas a lista de mais um nó conhecido. Esse processo é novamente iniciado partindo de N a solicitação de um novo vizinho a B. No momento da confirmação de recebimento é que um nó decide se deve buscar um novo vizinho.

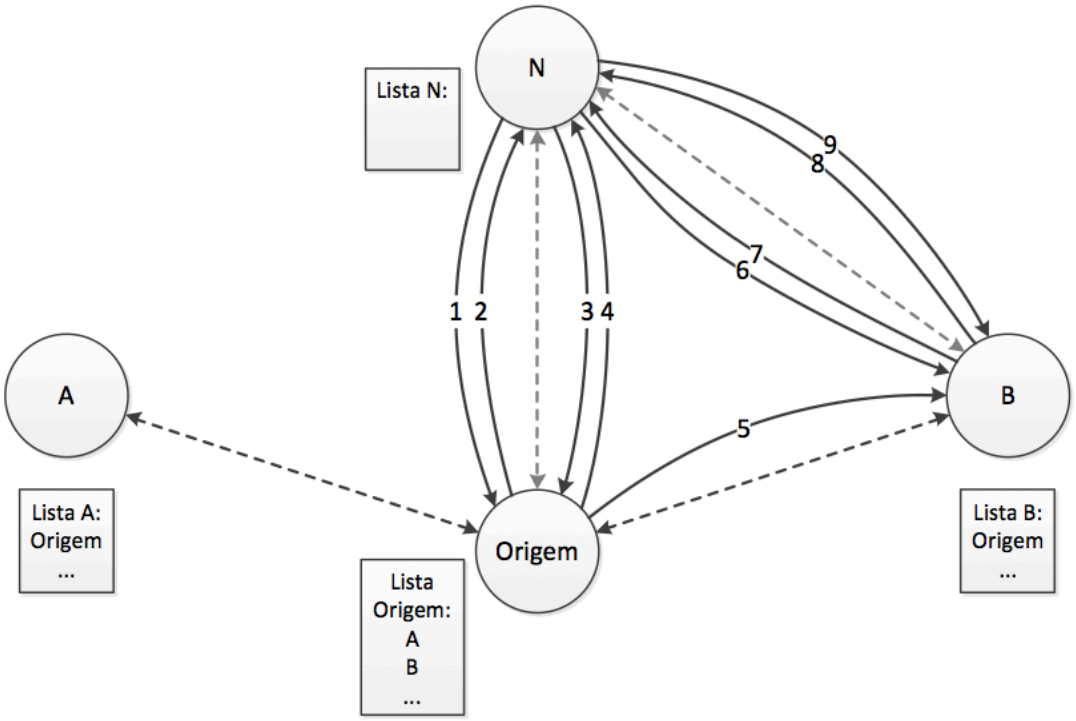

Figura 9 - Processo de descoberta de um vizinho [MORAES, H. F. et al., 2012a]

4.5.2 SOBRECARGA DE MENSAGENS

Para avaliar a sobrecarga de mensagens na utilização do protocolo REPI em uma rede, é importante levar em consideração dois fatores: a quantidade de mensagens de controle trocadas e a quantidade de mensagens propagadas colaborativamente.

Para se calcular a quantidade de mensagens de controle trocadas, usamos como base a quantidade de mensagens trocadas para descobrir um vizinho demonstrada na Seção anterior. Em [MORAES, H. F. et al., 2012a] é demonstrado o cálculo total de mensagens de controle trafegadas na rede. Esse cálculo leva em consideração a formação da vizinhança e a manutenção dos canais de comunicação dos vizinhos dos N nós presentes na rede.

O número total de mensagens (T) é o produto do número de mensagens trafegadas de um nó (M_n) e o número de nós presentes na rede (N), ou seja, $T = M_n * N$. O número de mensagens que um nó troca (M_n) é correspondente à soma de mensagens para estabelecer a vizinhança (M_v) mais o número de mensagens para manter esse canal de comunicação (M_{mc}), ou seja, $M_n = M_v + M_{mc}$. O número de mensagens para estabelecer a vizinhança (M_v) é o produto do número de

mensagens para obter um vizinho (7) e a quantidade de vizinhos de um nó (N_v), ou seja, $M_y = 7 * N_v$. Analogamente, o número de mensagens de manutenção (M_{mc}) é o produto do número de mensagens necessárias para manter uma conexão (M_{muc}) e a quantidade de vizinhos de um nó (N_v), ou seja, $M_{mc} = M_{muc} * N_v$.

Já sob a ótica das mensagens propagadas colaborativamente, [MORAES, H. F. et al., 2012a] destaca que a retransmissão das mensagens segue alguns critérios definidos em cada nó, e esses critérios são utilizados como ferramentas para impedir que as mensagens inundem a rede. Além disso, [MORAES, H. F. et al., 2012a] argumenta que o protocolo possui um campo de cabeçalho contendo a quantidade máxima de saltos (HTL - *Hop To Live*) bem como uma tabela contendo as últimas mensagens já enviadas. Tais mecanismos visam prevenir o tráfego indefinido de mensagens na rede tal como proporcionar uma economia de recursos, evitando encaminhamentos desnecessários.

Na Seção 6.3, é feita uma avaliação quantitativa em cenário típico de um ambiente ubíquo.

4.5.3 MIDDLEWARE DE SUPORTE E API DE DESENVOLVIMENTO

De forma a se implementarem os conceitos demonstrados nas seções anteriores, [MORAES, H. F. et al., 2012a] desenvolveram um middleware responsável por abstrair detalhes intrínsecos do protocolo. Este middleware permite não só a formação da vizinhança e dos canais de comunicação de forma transparente como também disponibiliza uma API de desenvolvimento para a criação de aplicações que utilizem o REPI como mecanismo de comunicação. Ele é o responsável por fazer a ligação entre as aplicações e o protocolo REPI.

O middleware possui suporte a Linux e Android e APIs de desenvolvimento em Java, Python e C/C++. O seu funcionamento se dá através de um *Daemon* instalado em cada dispositivo. As aplicações que desejam interagir com o protocolo estabelecem um Socket de comunicação com o *Daemon*. Este, por sua vez, trata de transmitir adequadamente as mensagens enviadas a este Socket seguindo as primitivas do protocolo.

O funcionamento no Android ocorre através da instalação do *Daemon* no dispositivo. Este *Daemon* é empacotado em uma aplicação Android e funciona como um serviço em segundo plano (veja a Figura 10). Entretanto, é necessário possuir

privilégios de superusuário para o *Daemon* poder configurar corretamente o dispositivo.

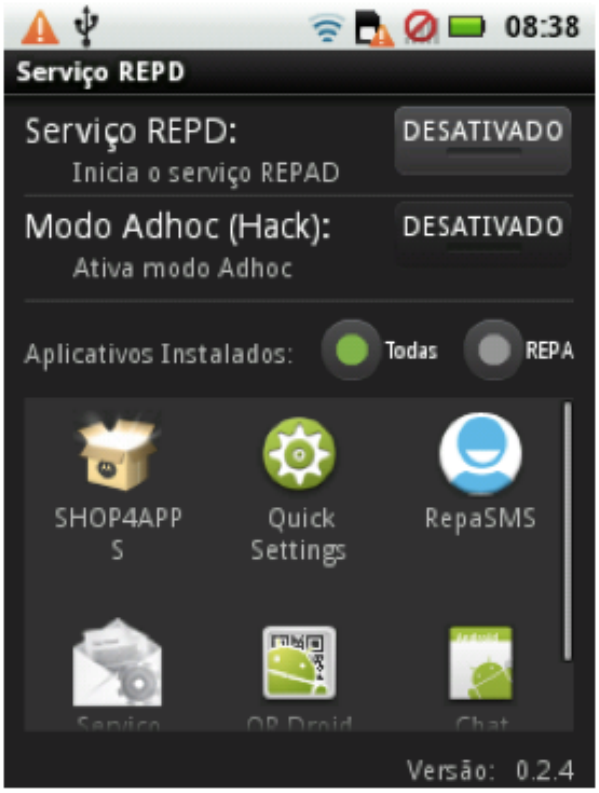

Figura 10 - Aplicação que empacota o Daemon de comunicação ação

Uma vez instalado o *Daemon* no Android, é possível configurar se o mesmo está ligado ou desligado e se o mesmo será utilizado em modo Ad-hoc ou não. Destaca-se que o funcionamento Ad-hoc só é possível através de um artifício de programação, uma vez que a plataforma Android não possui suporte oficial para a criação de redes Ad-hoc. Sendo assim, [MORAES, H. F. et al., 2012a] argumenta que este modo pode não funcionar em todos os dispositivos.

4.5.4 TRATAMENTO DE ERROS DE COMUNICAÇÃO

A versão atual da API desenvolvida por [MORAES, H. F. et al., 2012a] disponibiliza alguns métodos para o tratamento de situações não esperadas. Uma vez que o desenvolvimento se dá através da criação de Sockets de comunicação análogos aos tradicionais Sockets, nela é possível tratar se uma conexão está aberta e se houve algum problema. Além disso, é possível esperar pelo recebimento de uma mensagem durante um determinado tempo estabelecido quando se inicia

um canal de comunicação. Tais ferramentas permitem um maior controle no caso de eventos não esperados.

Entretanto, é importante destacar que, apesar de uma nova versão estar sendo desenvolvida com um suporte maior à garantia de entrega, a versão atual da API não garante a entrega de 100% das mensagens, ou seja, se houver uma falha na entrega da mensagem devido a um problema no *Daemon* ou na rede, ela não será reenviada. Apesar disso, é importante frisar que a taxa de entrega levantada por [MORAES, H. F. et al., 2012a] é superior a 96% e que em um ambiente infraestruturado, por exemplo, todos os nós estão conectados por um ponto de acesso, o que diminui o risco em caso de falha em algum nó.

Na Seção 6.3, é feita uma avaliação quantitativa em cenário típico de um ambiente ubíquo.

4.5.5 PRIVACIDADE E SEGURANÇA

Do ponto de vista de segurança, é importante destacar que o REPI é construído através de uma rede colaborativa. Sendo assim, riscos pertinentes a essa abordagem precisam ser levados em consideração. Se utilizado em uma rede infraestruturada, é recomendado o uso de protocolos de segurança como WPA.

A versão atual da API desenvolvida por [MORAES, H. F. et al., 2012a] disponibiliza também a possibilidade de enviar mensagens anonimamente. Isso pode ser potencialmente útil a aplicações mais sensíveis.

Diante disso, é recomendado que cada aplicação avalie individualmente os riscos associados ao conteúdo trafegado, para que mecanismos de segurança a nível de aplicação possam ser adotados conforme necessário.

4.5.6 ECONOMIA DE ENERGIA

Sob a ótica de economia de energia, um fator a se considerar é o gasto de energia devido a grande troca de mensagens. Entretanto, na Seção 4.5.2 pudemos constatar que o número de mensagens trocadas é previsível e não cresce indefinidamente.

Apesar disso, um ponto importante é o gasto gerado pelo *Daemon*. No caso de uso em modo Ad-hoc instalado em um dispositivo Android, o *Daemon* em curto espaço de tempo consome a bateria do dispositivo. Já no caso de uso de uma rede infraestruturada, o *Daemon* não gera impacto expressivo no consumo de energia.

Na Seção 6.3, é feita uma avaliação quantitativa em cenário típico de um ambiente ubíquo.

4.5.7 QUALIDADE DE SERVIÇO (QOS)

A versão atual do protocolo não permite nativamente o suporte à garantia de serviços diferenciados. Sendo assim, é necessário a nível de aplicação construir mecanismos que suportem tratamentos particulares.

4.5.8 ESCALABILIDADE DO PROTOCOLO

Em [MORAES, H. F. et al., 2012a] é demonstrado que o protocolo é escalável linearmente com a adição de novos nós. Além disso, diversos testes foram feitos utilizando redes com quantidade de nós variando de poucos ao extremo de 10240 nós. Experimentos também foram feitos variando a densidade da rede, e, em todos, a taxa de entrega e o tempo de resposta foram satisfatórios. Em [MORAES, H. F. et al., 2012a], diversos gráficos dos resultados podem ser verificados.

4.6 CONCLUSÃO DO CAPÍTULO

Este Capítulo apresentou o Protocolo de Comunicação Orientado a Interesses proposto por [DUTRA, R. DE C. et al., 2010]. O protocolo se diferencia dos demais por não utilizar o endereçamento origem-destino. Utilizando-se do mecanismo de interesses nas mensagens, [DUTRA, R. DE C. et al., 2010] baseia-se no modelo de *publish-subscribe* para permitir que os usuários troquem mensagens de acordo com os interesses assinados. Já [MORAES, H. F. et al., 2012a] propôs um modelo P2P, que cria uma rede sobreposta responsável por fornecer as características de funcionamento do protocolo criado na estrutura atual da internet.

Além disso, um middleware responsável por abstrair detalhes intrínsecos do protocolo é detalhado. Este middleware permite não só a formação da vizinhança e dos canais de comunicação de forma transparente como também disponibiliza uma API de desenvolvimento para a criação de aplicações que utilizem o REPI como mecanismo de comunicação. O protocolo também é detalhado em termos de tratamento de erros da comunicação, economia de energia, segurança, qualidade de serviço e escalabilidade.

CAPÍTULO 5 FRAMEWORK INTEGRADO AO REPI

5.1 INTRODUÇÃO

O Capítulo 3 descreveu o framework proposto por [MARELI, D. et al., 2013], enquanto o Capítulo 4 demonstrou o funcionamento e as características do protocolo proposto por [DUTRA, R. DE C. et al., 2010]. Estes trabalhos possuem semelhanças no que diz respeito aos mecanismos de comunicação.

Neste Capítulo, apresenta-se como a proposta de [DUTRA, R. DE C. et al., 2010] poderia ser utilizada como alternativa na implementação dos mecanismos de comunicação previstos pelo framework. Na Seção 5.2, identifica-se o que motivou a integração de ambas as propostas e na Seção 5.3, como o protocolo se integra com a identificação dos recursos. Na Seção 5.4, descreve-se como se dá a integração do protocolo com a Invocação Remota de Método e na Seção 5.5, como se dá a integração do protocolo com a Comunicação por Eventos ou Interesses. Na Seção 5.6 é apresentada a API de integração com o protocolo REPI. Na Seção 5.7 é descrito o estado da arte relacionado. Finalmente na Seção 5.8 é detalhado o funcionamento da nova API de integração com o protocolo REPI.

5.2 MOTIVAÇÃO

O funcionamento do protocolo abordado nas seções 4.3 e 4.4 demonstra que o protocolo está aderente aos conceitos de comunicação orientada a interesses implementados no Framework UFF [MARELI, D. et al., 2013] (veja a Seção 3.9.2). Desta forma, conforme já considerado na Seção 4.2, este protocolo poderia ser utilizado como uma alterativa de implementação dos mecanismos de comunicação previstos pelo framework.

Além disso, o suporte desenvolvido por [MORAES, H. F. et al., 2012a] para a utilização do protocolo na internet (veja a Seção 4.5.3) possibilita e facilita a integração do protocolo com a implementação do framework (veja a Seção 3.8). Em particular, o middleware desenvolvido por [MORAES, H. F. et al., 2012a] já fornece suporte à plataforma Android que também foi utilizada na implementação do framework (veja a Seção 3.8).

5.3 INTEGRAÇÃO DO REPI COM A IDENTIFICAÇÃO DOS RECURSOS

No modelo REPI integrado a SmartAndroid, cada Agente de Recurso possuirá um Prefixo Ativo (PA) composto de um Prefixo (P) e dos Interesses (I) deste agente. Na inicialização do AR, os interesses básicos do agente são devidamente registrados, porém, durante o seu ciclo de vida, novos interesses podem ser adicionados ou removidos.

Na Figura 11, tem-se um AR fogão com suas variáveis de contexto e com o seu Prefixo (P). A seta do passo 1 representa a subscrição inicial que o AR Fogão fez quando foi inicializado. Se dinamicamente o ambiente agora dispõe de uma TV e um Smartphone, o AR poderá registrar interesse a qualquer momento nesses ARs (passos 2 e 3), ou seja, os interesses do AR podem mudar constantemente. O mesmo vale para a lista de agentes interessados no AR Fogão. No passo 4 é possível observar um novo interessado no AR Fogão registrando interesse na variável de contexto “setTimer”.

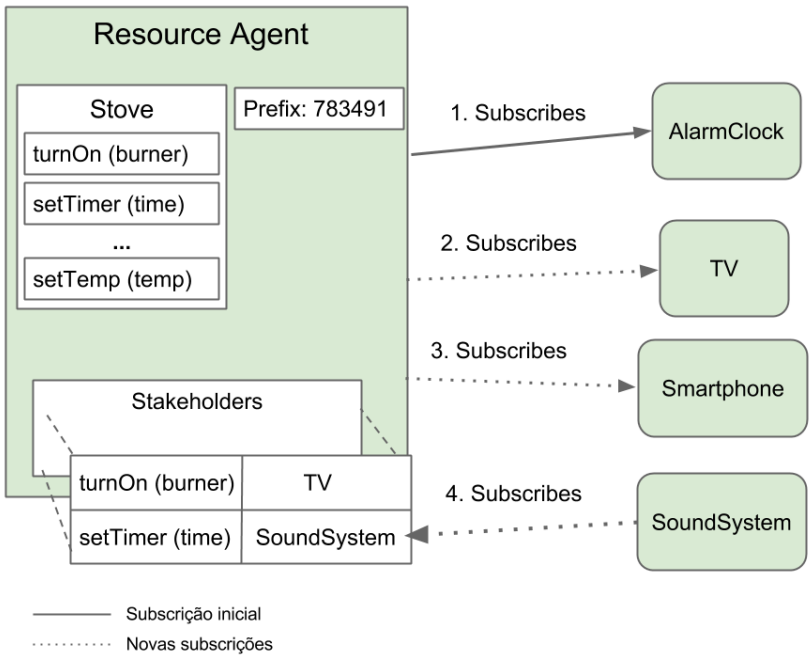

Figura 11 – AR Fogão com suas variáveis de contexto e fazendo e recebendo registros de interesse

Para viabilizar a integração do REPI a SmartAndroid, o primeiro passo é ampliar o uso do Sistema de Nome de Agentes de Recurso (RANS), abordado na Seção 3.9.3, para ter a opção de utilização da infraestrutura de encaminhamento do protocolo REPI. Para isso, é incluído um campo prefixo na resposta a uma consulta

de resolução de nome. No momento do registro do agente no Serviço de Registro de Recurso (SRR), onde apenas era informado o IP de localização do agente, passa a informar-se também o campo Prefixo (P) do Prefixo Ativo do agente. Assim sendo, uma consulta ao nome “lampadadasala.ra”, que antes retornava somente o <IP> do dispositivo onde o agente está, passa a retornar um objeto com um par <IP, Prefixo>.

A ampliação feita no RANS, para o protocolo REPI, foi projetada para possibilitar a adição de novos protocolos ao framework. A estrutura de dados retornada permite que sejam incluídas, no registro e na resposta de uma consulta ao RANS, informações relevantes para outros protocolos. Para prevenir o tráfego de informações de protocolos que não estão sendo utilizados no momento destaca-se a importância de se desenvolver uma API genérica que entregue somente os atributos do(s) protocolo(s) utilizado(s).

O campo Prefixo (P) é capaz de identificar unicamente o dispositivo em que o agente está. Isso permite associar um nome a um dispositivo específico, assim como permite a troca de mensagens fim-a-fim entre Agentes de Recurso, utilizando a infraestrutura do REPI. Veja na Figura 12 o novo funcionamento do RANS.

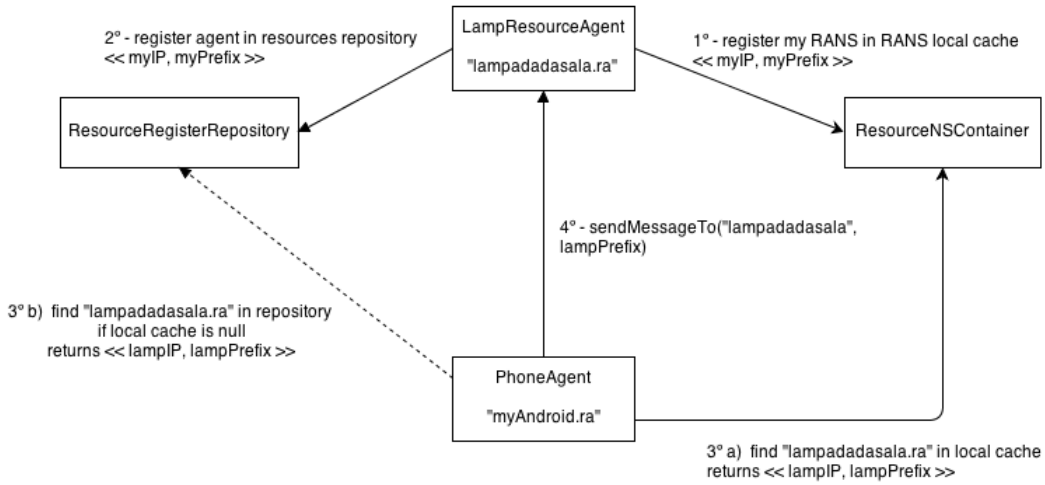

Figura 12 – RANS – Integrado com o REPI

A versão da API de integração com o protocolo REPI (REPA) utilizada na integração com a SmartAndroid somente permite a criação de um único socket de comunicação por aplicação com a infraestrutura da rede. Isso significa que somente um Prefixo Ativo pode existir por aplicação, o que na prática, no caso da SmartAndroid, limita a criação de um único agente por aplicação. Além disso, se

dois agentes de uma mesma aplicação quisessem se comunicar, não seria possível, pois o *Daemon* dessa versão não entregaria a mensagem, visto que o Prefixo de origem e destino seriam o mesmo. Ao final deste Capítulo, na Seção 5.8, é discutido como a versão mais recente da API (versão 1.2) facilita essa integração, como ela se adequaria a esta proposta e o que ainda precisa ser gerenciado a nível de aplicação.

Para contornar essas limitações, foi construída uma infraestrutura de abstração de integração com o protocolo REPI. Esta infraestrutura permite a criação virtual de mais de um PA por dispositivo. Assim sendo, todos os agentes presentes em uma aplicação possuem o mesmo campo Prefixo (P), porém esta infraestrutura é capaz de encaminhar corretamente as mensagens ao Agente de Recurso adequado. Veja na Figura 13 um exemplo de duas aplicações distintas de dois dispositivos, seus respectivos agentes e Prefixos Ativos (PA).

Essa infraestrutura é constituída de um *daemon* de recebimento e processamento de mensagens REPI (CommREPAD) mais uma API de abstração de comunicação com o protocolo REPI. Esta API estende a atual API de comunicação, fornecendo de forma transparente e unificada, as mesmas funcionalidades já empregadas na API atual. Desta forma, é possível trocar facilmente a implementação utilizada através de uma configuração. Além da interface padrão, esta nova API é capaz de fornecer mecanismos de comunicação não antes presentes na API atual. Disponibilizando, assim, métodos que se beneficiem de todas as vantagens que o protocolo REPI pode oferecer. O uso desta API, bem como as vantagens práticas oferecidas pelo protocolo, serão mais aprofundados na Seção 5.6.

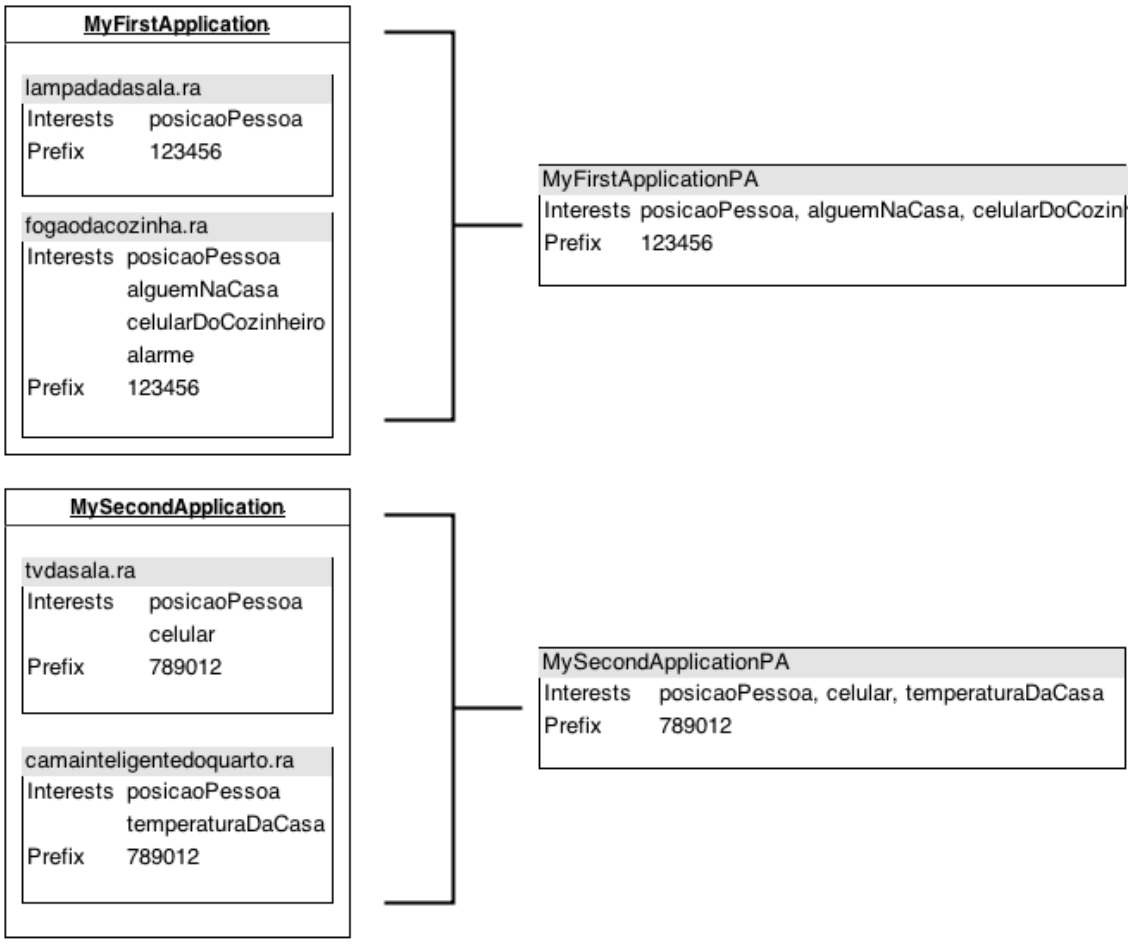

Figura 13 – Exemplo de duas aplicações distintas de dois dispositivos, seus respectivos agentes e Prefixos Ativos (PA)

Funcionando de maneira análoga a um servidor web, o *daemon* CommREPAD é responsável por receber todas as mensagens REPI. Ao receber uma nova mensagem, uma nova sessão de tratamento dessa mensagem (REPASession) é iniciada. Após sua inicialização, o *daemon* volta a escutar novas mensagens. A REPASession, por sua vez, trata a mensagem recebida, descomprimindo-a e invocando o método da API responsável por entregar a mensagem para os Agentes de Recurso interessados.

De forma a ser possível encaminhar adequadamente as mensagens que chegam à aplicação SmartAndroid aos devidos Agentes de Recurso interessados, é necessário estabelecer duas estruturas de mapeamento. A primeira, “agentsInterestsIndex”, utilizada quando a mensagem destina-se a um AR específico, possui a formação << rans + “://:” + interest, list(56all-back56) >>. Já a segunda, “myInterests”, utilizada quando a mensagem não é destinada a nenhum

AR específico, possui a formação << interest, list(57all-back57) >>. Essas estruturas são alimentadas no momento em que um AR registra um interesse.

Desta forma, se um agente de recurso “lampadadasala.ra” registrar interesse em “pessoaNaSala” com um 57all-back “meuCallbackASerExecutado”, a primeira estrutura ficará << “lampadadasala.ra://:pessoaNaSala”, “meuCallbackASerExecutado” >>; já a segunda ficará << “pessoaNaSala”, list(“meuCallbackASerExecutado”) >>.

A representação nessas duas estruturas permite que, quando chegue uma mensagem ao agente “lampadadasala.ra” no interesse “pessoaNaSala”, o 57all-back “meuCallbackASerExecutado” seja executado. Da mesma maneira, caso chegue uma mensagem apenas encaminhada para o interesse “pessoaNaSala”, não só o 57all-back “meuCallbackASerExecutado”, como qualquer outro 57all-back registrado no interesse “pessoaNaSala” seja executado.

5.4 INTEGRAÇÃO DO REPI COM A INVOCAÇÃO REMOTA DE MÉTODO

Conforme já destacado nas seções 3.5.1 e 3.9.1, a invocação remota de método é utilizada para executar uma ação síncrona no agente destino, sendo caracterizada por uma chamada a um método existente no agente destino, permitindo que as invocações locais feitas aos métodos sejam refletidas em invocações remotas de forma transparente.

Para se permitir a invocação remota de método utilizando-se o protocolo Json-RPC (veja a Seção 3.9.1), foi desenvolvido, na estrutura de comunicação REPI, o suporte para execução síncrona de métodos. Através de métodos bloqueantes, é possível uma mensagem enviada aguardar uma mensagem de resposta.

Visando à clara separação entre a camada de comunicação e o mecanismo de execução remota de métodos, todo Agente de Recurso interessado em receber um procedimento remoto registra um interesse “jsonrpc” com um 57all-back “JSONRPCCallback”. Este 57all-back é responsável por delegar a execução para o Dispatcher definido pelo protocolo Json-RPC. Destaca-se que o mesmo Dispatcher atualmente usado na SmartAndroid é utilizando em ambas as formas de comunicação e que a classe pai de todos os agentes “ResourceAgent” abstrai esse procedimento de registro de interesse em “jsonrpc”, simplificando o desenvolvimento.

No diagrama de classes da Figura 14 é possível identificar as principais classes envolvidas na integração da SmartAndroid com o protocolo REPI.

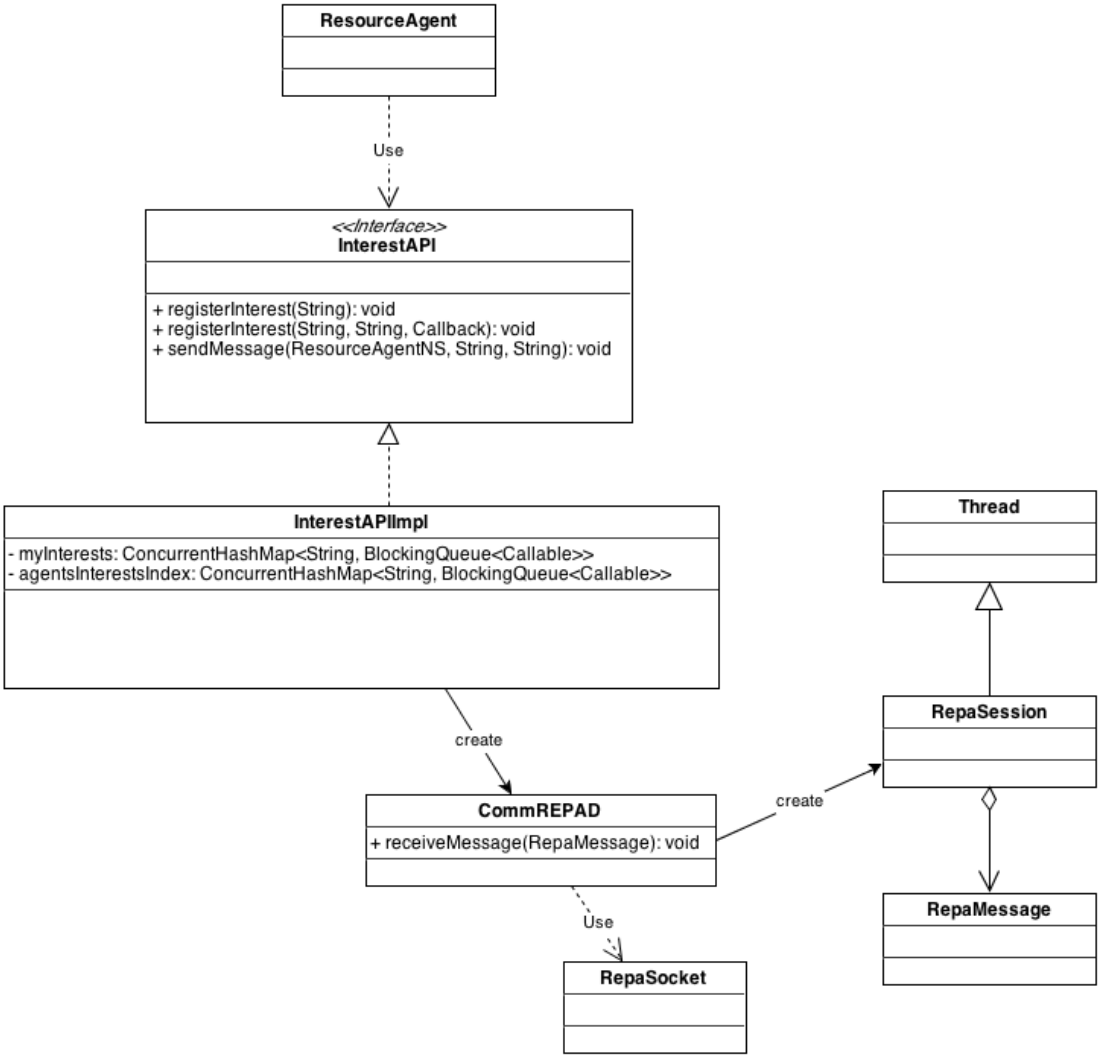

Figura 14 – Classes envolvidas na integração da SmartAndroid com o protocolo REPI

5.5 INTEGRAÇÃO DO REPI COM A COMUNICAÇÃO POR EVENTOS OU INTERESSES

Conforme já destacado nas seções 3.5.2 e 3.9.2, a comunicação por eventos ou interesses é utilizada quando um agente de recurso deseja ser notificado de alterações em alguma variável de contexto do agente destino. Para isso, esse agente subscreve-se em uma variável de contexto, informando estar interessado nas alterações que ocorrem nesta variável. Quando ocorrer uma alteração na variável de contexto subscrita, os agentes interessados nesta variável serão imediatamente notificados.

Para permitir esse tipo de comunicação, é utilizada uma API de abstração, citada na Seção 5.3, que estende a atual API de comunicação, fornecendo de forma transparente e unificada, as mesmas funcionalidades já empregadas na API atual. Esta API permite que um Agente de Recurso tanto registre o interesse em uma variável de contexto quanto seja notificado de alterações nesta variável. A Seção a seguir descreve mais detalhes desta API de comunicação.

5.6 API DE INTEGRAÇÃO COM O PROTOCOLO REPI

Conforme citado na Seção anterior, a API de comunicação atual da SmartAndroid e a integrada ao REPI possuem a mesma interface básica de uso. Entretanto, o uso do REPI traz alguns benefícios que hoje não estão disponíveis na comunicação atual da SmartAndroid, como a listagem de Agentes de Recursos usando apenas a infraestrutura da rede em caso de falha do repositório de recursos ou até mesmo a criação de redes sobrepostas para troca de conteúdo entre diversos dispositivos que adotam um padrão comum de interação. Sendo assim, além de ser possível interagir com uma API única, também é possível utilizar métodos específicos que o uso do protocolo REPI possibilita.

A Figura 15 mostra todos os métodos disponíveis na API de comunicação integrada ao REPI. Nela destacam-se os métodos provenientes da integração com o REPI: “sendMessage(String interest, String message)”, onde é possível enviar uma mensagem apenas especificando o interesse e o conteúdo da mensagem, bem como os métodos: “registerInterest(String interest, Callable 59all-back)”, o qual permite registrar um interesse, já associando um 59all-back a ser executado quando uma mensagem for recebida nesse interesse; o método “getListOfResourceAgents()”, o qual é capaz de listar todos os Ars presentes no sistema utilizando apenas a infraestrutura da rede; o método “getListOfResourceAgentsInterestedIn(String interest)”, que lista todos os Ars interessados em determinado interesse e, por fim, o método “getListOfInterestedIn(String interest)”, que lista todos os interessados em determinado interesse. Os demais métodos são de implementação direta em ambas as abordagens.

```
public interface InterestAPI {
    public void registerInterest(String rans) throws Exception;
    public void registerInterest(String rans, String interest, Callable callback) throws Exception;
    public void registerInterest(String interest, Callable callback) throws Exception;
    public String sendMessage(ResourceAgentNS raNSfrom, ResourceAgentNS raNSto, String interest, String message) throws Exception;
    public String sendMessage(ResourceAgentNS raNSfrom, Integer prefixTo, String interest, String message) throws Exception;
    public String sendMessage(ResourceAgentNS raNSto, String interest, String message) throws Exception;
    public String sendMessage(Integer prefixTo, String interest, String message) throws Exception;
    public String sendMessage(String interest, String message) throws Exception;
    public void sendAsyncMessage(ResourceAgentNS raNSfrom, ResourceAgentNS raNSto, String interest, String message, Callable callback) throws Exception;
    public void sendAsyncMessage(ResourceAgentNS raNSfrom, int prefixTo, String interest, String message, Callable callback) throws Exception;
    public void sendAsyncMessage(ResourceAgentNS raNSto, String interest, String message, Callable callback) throws Exception;
    public void sendAsyncMessage(int prefixTo, String interest, String message, Callable callback) throws Exception;
    public List<ResourceAgentNS> getListOfResourceAgents() throws Exception;
    public List<ResourceAgentNS> getListOfResourceAgentsInterestedIn(String interest) throws Exception;
    public List<String> getListOfInterestedIn(String interest) throws Exception;
}
```

Figura 15 – API de Comunicação Integrada ao REPI

5.7 ESTADO DA ARTE

A fim de se aprofundar na área e de modo a auxiliar o desenvolvimento e comparação da abordagem desenvolvida, diversos trabalhos na área de computação ubíqua, ambientes inteligentes, *smarthomes* e simuladores foram estudados. Estes trabalhos foram analisados sob a ótica de como a comunicação se dá nas propostas desenvolvidas por eles.

Os trabalhos que já utilizam o REPI para a troca de mensagens ainda estão em desenvolvimento, porém é possível citar trabalhos em andamento com sensoriamento de temperatura em caixa de abelha, feito pela Universidade Estadual de Maringá (UEM); na área de segurança da informação, feito pelo Laboratório Nacional de Computação Científica (LNCC); sensoriamento de incêndio, comunicação em redes veiculares e comunicação em veículos não tripulados (VANTS) desenvolvidos pelo Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia (COPPE-UFRJ).

Em [DEY, A. et al., 2001], a comunicação se dá através da plataforma TCP/IP e pelo uso de um mecanismo simples de comunicação baseado em HTTP e em mensagens XML codificadas. Tal escolha leva em consideração questões de portabilidades oferecidas pela pilha TCP/IP. É destacada a preocupação com a escalabilidade do sistema com um aumento na quantidade de dispositivos, porém nos testes realizados, o limite da plataforma não foi atingido. Assim como esta proposta, o protocolo de comunicação foi projetado para ser plugável. De forma análoga, [DEY, A. et al., 2001] também utiliza um *Daemon* para o recebimento dos eventos e destaca a importância de um tratamento diferenciado para *streams* de conteúdo enviados constantemente. Por exemplo, um sensor de temperatura notifica a mudança de temperatura com variação de 0,1 graus Celsius, visando evitar um possível gargalo gerado pela estrutura de processamento de eventos.

Em [KIM, I. Et al., 2006], a comunicação é feita através da tecnologia *JavaSpaces,* a qual permite a comunicação remota entre aplicações distribuídas. Um espaço virtual de troca de informações é criado, e nele permite-se a troca de objetos e dados abstraindo problemas típicos que um ambiente distribuído traz. Um gerenciador de eventos é utilizado para o controle de eventos produzidos por sensores. Nenhuma API genérica de comunicação é apresentada bem como alternativas para a troca do mecanismo de comunicação utilizado.

Em [ARMAC, I. Et al., 2007], a comunicação é dividida em dois tipos: a comunicação entre os objetos presentes no simulador e a comunicação entre os dispositivos e o simulador. Entre os objetos presentes no simulador, a troca de informações ocorre através de mudanças de estados observadas pelos seus interessados. Já a comunicação entre os dispositivos e o simulador se dá através de métodos fornecidos pelos drivers dos dispositivos. Em termos gerais, a arquitetura de comunicação é bastante semelhante à adotada na SmartAndroid, porém o funcionamento da comunicação em termos de encaminhamento e API de uso, não são abordados.

Em [VAN NGUYEN, T. Et al., 2009] a comunicação é orientada a conexões remotas entre hosts presentes na rede. Destaca-se a necessidade de integração de um conjunto de funções tanto no servidor quanto no cliente para o estabelecimento dessa comunicação. Para o registro e descoberta de recursos um servidor web é utilizado.

Em [RANGANATHAN, A. et al., 2005], a comunicação se dá de fato seguindo o padrão CORBA. O framework Olympus proposto é capaz de descobrir a melhor entidade que satisfaz determinados requisitos. A abordagem da SmartAndroid, através do Serviço de Descoberta de Recursos (SDR), também é capaz de descobrir a melhor entidade que satisfaz determinados requisitos.

Em [FILLINGER, A. et al., 2009], a comunicação é voltada para o processamento de *streams* de dados provenientes de multisensores. Para tal, foram desenvolvidos protocolos de transporte de dados com o mínimo de overhead possível. Além disso, sempre que possível, a troca de mensagens se dá através de *shared memory*. Os protocolos desenvolvidos em [FILLINGER, A. et al., 2009] poderiam ser acoplados à API de comunicação de forma que, de acordo com o cenário, o protocolo mais adequado seja escolhido pela API.

5.8 NOVA API DE INTEGRAÇÃO COM O PROTOCOLO REPI

Conforme citado na Seção 5.3, a versão da API utilizada para a integração com o protocolo REPI possui algumas limitações, dentre elas, a impossibilidade de ter mais de um Socket de comunicação com o *Daemon* e a impossibilidade de enviar uma mensagem entre dois agentes dentro de uma mesma aplicação ou dispositivo, visto que o prefixo de origem e destino são os mesmos. Neste caso, o *Daemon* preveniria a entrega da mensagem.

Entretanto, há uma nova versão da API de comunicação com o protocolo REPI que permite a criação de mais de um Socket de comunicação com o *Daemon*. Além disso, cada Socket possui seus respectivos interesses, e agora o *Daemon* permite que mensagens sejam trocadas internamente a uma aplicação, desde que os Sockets sejam distintos.

Com essa nova versão da API, é possível fazer otimizações na integração da SmartAndroid com a API do REPI. Visto que nesta versão é possível estabelecer mais de um Socket de comunicação com o *Daemon*, é possível reduzir a infraestrutura citada na Seção acima. Sendo assim, cada Agente de Recurso poderia ficar responsável por estabelecer um canal de comunicação com o *Daemon*.

Evidentemente todo esse suporte já estaria disponível na classe abstrata de um Agente de Recurso. Como os interesses estão segregados por Socket de comunicação, todos os agentes teriam seus interesses desacoplados de forma transparente graças à nova infraestrutura fornecida. No entanto, para o correto funcionamento de uma comunicação direta entre dois agentes, é fundamental que seja identificado e validado o RANS de destino. Essa verificação é fundamental, pois podem existir dois agentes com os mesmos interesses, e ambos receberão a mensagem. Essa validação deve ser provida pela classe abstrata de um Agente de Recurso.

Diante do fato de, nesta versão, o *Daemon* permitir trocas de mensagens internas, a comunicação entre os Agentes de Recursos poderia ser efetuada transparentemente. O *Daemon* efetuará o casamento do Prefixo mais os Interesses.

Vale destacar que a extensão da consulta do RANS como, por exemplo, “lampadadasala.ra” para retornar tanto o IP quanto o Prefixo ainda se faz necessária visto que dois agentes estabelecem uma comunicação entre si através de seus respectivos nomes, e o protocolo é usado apenas para o encaminhamento das mensagens. O suporte à execução síncrona de métodos criado também se faz necessário, visto que a API é toda assíncrona.

5.9 CONCLUSÃO DO CAPÍTULO

Neste Capítulo, foi discutida a utilização da proposta de [DUTRA, R. DE C. Et al., 2010] como alternativa na implementação dos mecanismos de comunicação previstos pelo framework. O Sistema de Nome de Agentes de Recurso (RANS) foi alterado para ter a opção de utilização da infraestrutura de encaminhamento do

protocolo REPI. Foi construída uma infraestrutura de abstração de integração com o protocolo REPI.

Para suportar a invocação remota de método foi desenvolvido, na estrutura de comunicação REPI, o suporte para a execução síncrona de métodos. Para suportar a comunicação por eventos ou interesses, foi utilizada uma API de abstração que estende a atual API de comunicação, fornecendo de forma transparente e unificada, as mesmas funcionalidades já empregadas na API atual.

Este Capítulo levantou o estado da arte relacionado e como a nova API de integração com o protocolo REPI se adequaria à proposta atual.

CAPÍTULO 6 AVALIAÇÃO

6.1 CRITÉRIO

Para avaliar a qualidade e a contribuição do trabalho, propõe-se o uso de questões de competência, como definido em [GRÜNINGER, M. et al., 1995]. Uma metodologia similar à descrita em [GRÜNINGER, M. et al., 1995] foi adotada para avaliar como o modelo de comunicação proposto por [DUTRA, R. DE C. Et al., 2010] e adaptado para a internet por [MORAES, H. F. et al., 2012a] se adequa a um ambiente ubíquo e inteligente. Através dessa metodologia, é possível determinar as vantagens e desvantagens que a adoção do protocolo REPI traz para o Framework UFF e para a infraestrutura da SmartAndroid.

Estas questões representam requisitos de aplicações que devem ser atendidos, utilizando os conceitos e mecanismos associados ao Framework UFF. Segundo consta em [GRÜNINGER, M. et al., 1995], quatro etapas são requeridas para a avaliação por questões de competência: (i) definição dos requisitos ontológicos em forma de questões que a proposta deve responder; (ii) definir terminologia da ontologia (elementos como objetos, atributos e relacionamentos), especificando a linguagem de expressão dos requisitos requeridos pela aplicação; (iii) usar a terminologia para representar a solução da aplicação; e (iv) testar a solução em termos das competências citadas, provando formalmente sua eficácia.

No contexto desta dissertação, a avaliação da adoção do protocolo REPI substitui cada etapa da seguinte forma: em (i) as questões são elaboradas textualmente; em (ii) são definidas representações das questões de competência em termos do Framework UFF; em (iii) são propostas duas resoluções através do Framework UFF da aplicação utilizada como exemplo, e em (iv) estas instâncias de competência (definidas na etapa (ii)) são executadas em um cenário de uso da aplicação. Destaca-se que na etapa (iii), uma resolução se dá através da implementação original da SmartAndroid, enquanto a outra resolução utiliza a infraestrutura do protocolo REPI.

6.2 APLICAÇÕES

Conforme discutido na Seção 5.3 e 5.6, a API de comunicação do REPI é uma extensão da API atual e possui a mesma interface básica de uso. Diante disso,

um exemplo simples, como o funcionamento de uma regra de contexto, como a exposta na Seção 3.6, não terá mudanças em relação a implementação utilizada. A regra “SE uma pessoa vai dormir E esquece o fogão ligado ENTAO dispara um alarme sonoro.” Pressupõe que as informações de contexto “posição da pessoa”, “estado da boca do fogão” e “alarmes disponíveis no ambiente” possam ser obtidas através da API de comunicação. Essas informações podem ser obtidas da mesma forma via API de comunicação, ou seja, não representam para o desenvolvedor uma mudança significativa.

Portanto, a implementação de diversas aplicações simples levam a conclusão que, para aplicações típicas, a implementação atual da SmartAndroid e a utilizando o protocolo REPI são equivalentes. Desta forma, as aplicações a seguir demonstram aspectos particulares que merecem ser destacados e analisados.

É importante destacar que este trabalho concentra-se nas questões de competência que envolvem soluções que estejam relacionadas à comunicação entre os dispositivos. Desta forma, para cumprir a primeira etapa da avaliação (i – elaboração textual), são enumeradas as seguintes questões:

1. Qualquer agente pode localizar agentes que atendam a determinado critério na presença de uma falha temporária do Repositório de Recursos?
2. Qualquer agente é capaz de lidar quando um conjunto de interessados em suas variáveis de contexto muda dinamicamente?
3. Qualquer agente é capaz de lidar quando é preciso enviar uma mensagem a um grande número de agentes interessados?
4. Qualquer agente pode criar uma rede sobreposta entre agentes com interesses em comum?

No restante deste Capítulo serão apresentadas as etapas (ii), (iii) e (iv), para demonstrar a utilização e validar, através da implementação de aplicações ubíquas, as questões de competência apresentadas. As aplicações descritas neste Capítulo foram implementadas e executadas sobre a plataforma Android, através da infraestrutura da SmartAndroid.

6.2.1.1 APLICAÇÃO DE MONITORAMENTO DE ACIDENTES COM IDOSOS

Esta Seção aborda a questão 1, relacionada ao agente poder localizar agentes que atendam a determinado critério na presença de uma falha temporária do Repositório de Recursos. Essa questão é explorada através do desenvolvimento de uma aplicação de monitoramento de acidentes com idosos.

Esta aplicação consiste em detectar, através de sensores, que um idoso sofreu um acidente de forma que se possa alertar as demais pessoas presentes na casa, ou, até mesmo, um assistente remoto, do ocorrido. Tal tipo de aplicação é relevante em situações em que o idoso esteja sozinho em determinado cômodo da casa e sofra algum tipo de acidente, como uma queda.

Cumprindo a etapa (ii), a questão 1, em termos do Framework UFF, pode ser representada através de um AR da aplicação, pelo Repositório de Recursos após sofrer uma falha não esperada, e pela ação do AR precisar localizar agentes que atendam a determinado critério. A Figura 16 demonstra essa interação.

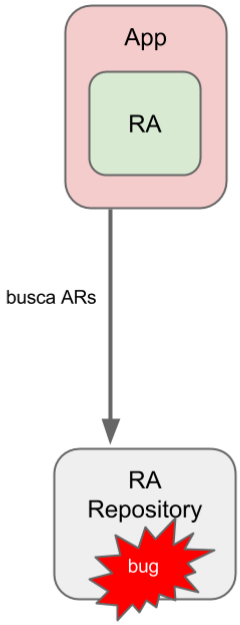

Figura 16 - Representação da questão de competência 1

Cumprindo a etapa (iii), a aplicação de monitoramento de acidentes é modelada na Figura 17, que descreve os principais componentes envolvidos, sendo eles: um AR sensor capaz de identificar um acidente, um AR da aplicação, o Repositório de Recursos e os diversos ARs capazes de informar o acidente ocorrido.

O cenário de uso da aplicação sem falhas é composto por uma sequência de passos. O AR sensor identifica um acidente e informa ao AR da aplicação (passo 1). O AR da aplicação busca no Repositório de Recursos agentes com capacidade de notificação que estejam no mesmo cômodo que as demais pessoas da casa (passo 2). O Repositório de Recursos devolve a lista de ARs que atendam a esse critério (passo 3). E por fim, o AR de monitoramento notifica os ARs localizados que houve um acidente (passo 4). Já no cenário previsto pela questão 1, ou seja, onde há uma falha no Repositório de Recursos, o passo 2 é prejudicado.

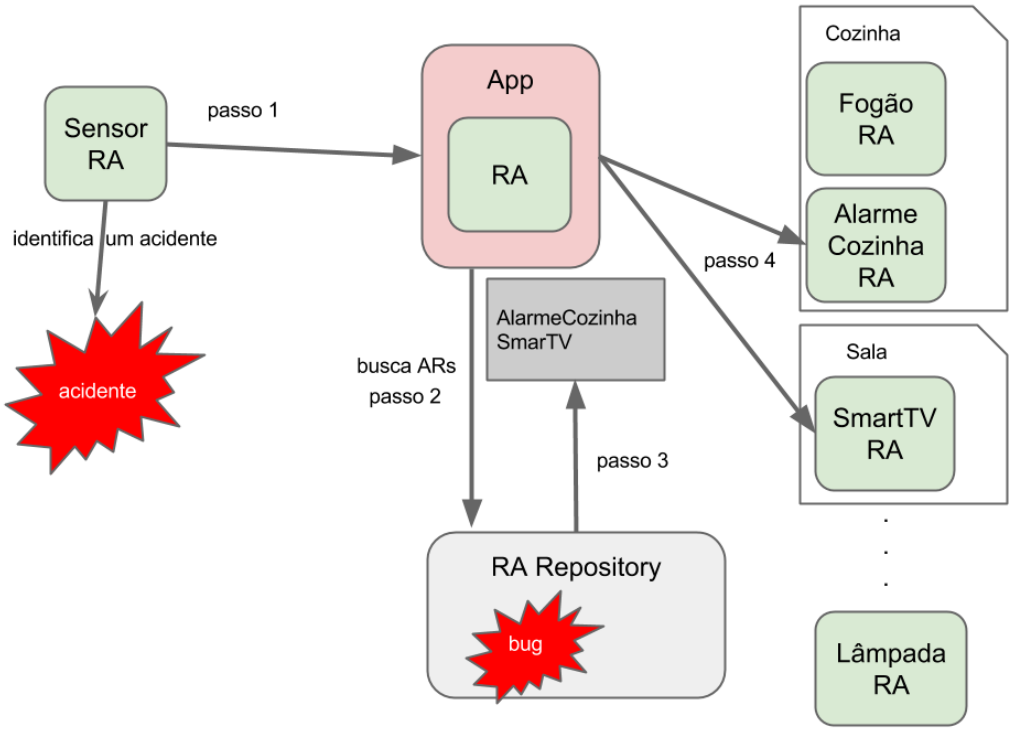

Figura 17 - Aplicação de Exibição ou Audição de Conteúdo em Dispositivos Móveis, Televisões e Rádios

Segundo a API de comunicação, o passo 2 é implementada através do método “getListOfResourceAgentsInterestedIn”, que lista todos os agentes que possuam um interesse específico como, por exemplo, “smartandroid://alarme”. O código detalhado é apresentado na Figura 18.

```
InterestAPI ia = InterestAPIImpl.getInstance();
try {
    String interest = "smartandroid://alarme";
    List<ResourceAgentNS> list = ia.getListOfResourceAgentsInterestedIn(interest);

    String message = "Alarmar - Acidente com um idoso - Help!!!";
    for (ResourceAgentNS resourceAgentNS : list) {
        ia.sendMessage(raNSFrom, resourceAgentNS, interest, message);
    }
} catch (Exception e) {
    System.out.println(e);
}
```

Figura 18 - Implementação de um alerta gerado por uma aplicação de monitoramento de idosos

Neste cenário de queda do Repositório de Recursos, a implementação utilizando o protocolo REPI é capaz de fazer a consulta diretamente na infraestrutura da rede, disparando diretamente a mensagem na rede ao invés de uma consulta específica ao Repositório de Recursos.

Destaca-se que a API usada pelo projetista da aplicação é a mesma em ambas as abordagens. Desta forma, o uso é o mesmo, porém internamente a implementação utilizando o REPI é capaz de entregar o resultado mesmo que o Repositório de Recursos esteja indisponível no momento.

Sendo assim, o código da Figura 18 poderia representar a solução em ambas as abordagens. Porém, como dito anteriormente, a implementação via REPI é capaz de lidar melhor com uma falha no Repositório de Recursos quando os agentes precisam ser encontrados dinamicamente, pois os interesses estão distribuídos pela rede. Questões relativas ao tratamento de erros da API do protocolo REPI são analisadas na Seção 4.5.4.

Em relação à etapa (iv), imagina-se um cenário de uma casa inteligente composta por uma família de pai, mãe, avó e crianças. Neste cenário, a avó está no quarto, enquanto o Pai está na sala, a Mãe, na cozinha e as crianças, brincando na varanda. Após a avó sofrer uma queda no quarto, um sensor do quarto identificará o acidente e notificará a aplicação de monitoramento de acidentes do ocorrido. A aplicação tentará buscar os ARs adequados para a notificação, porém o Repositório de Recursos, nesse momento, está indisponível devido a uma falha de software, por exemplo. Neste caso, a aplicação utilizando o protocolo REPI seria capaz de enviar uma mensagem para a rede através do interesse “smartandroid://alarme”,

solicitando que os dispositivos com capacidade de aviso localizados em cômodos com pessoas, caso da sala e cozinha, sejam notificados. Neste caso, a Smart TV presente na sala poderia informar o pai do ocorrido, e o alarme presente na cozinha poderia disparar um alerta sonoro.

6.2.1.2 APLICAÇÃO DE EXIBIÇÃO OU AUDIÇÃO DE CONTEÚDO EM DISPOSITIVOS MÓVEIS, TELEVISÕES E RÁDIOS

Esta Seção aborda a questão 2, relacionada ao agente ser capaz de lidar quando um conjunto de interessados em suas variáveis de contexto muda dinamicamente. Essa questão é explorada através do desenvolvimento de uma aplicação de exibição ou audição de conteúdo em dispositivos móveis, televisões e rádios de acordo com a localização da pessoa.

Esta aplicação é composta de diversos sensores responsáveis por identificar onde uma determinada pessoa está e de um conjunto de dispositivos que podem exibir um conteúdo de vídeo ou áudio. Pode-se reproduzir o conteúdo desejado nos dispositivos ao redor de uma pessoa, sendo a aplicação executada em smartphones, tablets, notebooks, computadores, televisões, entre outros. Neste cenário, diversos dispositivos de exibição de conteúdo estariam interessados na posição do AR Pessoa. Vale destacar que a transmissão efetiva de um vídeo ou áudio requer protocolos específicos para este fim.

Cumprindo a etapa (ii), a questão 2, em termos do Framework UFF, pode ser representada através de um AR Pessoa e de diversos ARs da aplicação interessados na posição do AR Pessoa. Quando esse AR Pessoa muda do quarto para a sala, por exemplo, os ARs de recurso dessa aplicação presentes na sala registrarão interesse na posição do AR Pessoa. Caso ocorra uma mudança na posição do AR Pessoa, os novos ARs interessados serão notificados. A Figura 19 demonstra essa interação.

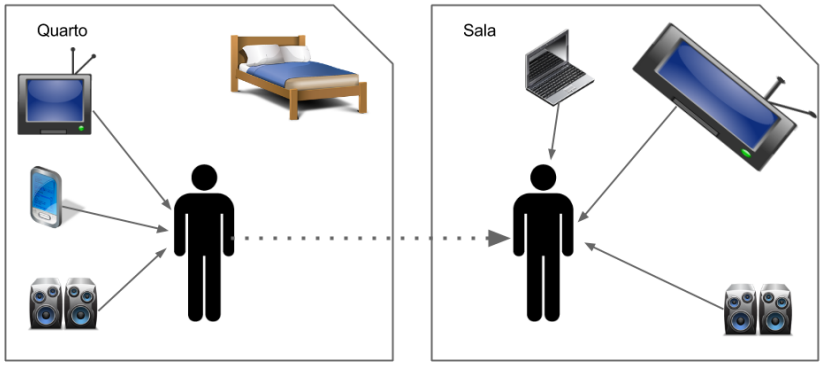

Figura 19 - Representação da questão de competência 2

Cumprindo a etapa (iii), a aplicação de exibição de conteúdo é modelada na Figura 20. A Figura 20 descreve os principais componentes envolvidos, sendo eles: um AR Pessoa e dois ambientes (sala e quarto) com seus respectivos ARs. Inicialmente o AR Pessoa está no quarto com os ARs presentes neste ambiente interessados na sua posição (passo 1). Quando o AR Pessoa se desloca para a sala (passo 2), os ARs presentes na sala registrarão interesse no AR Pessoa (passo 3).

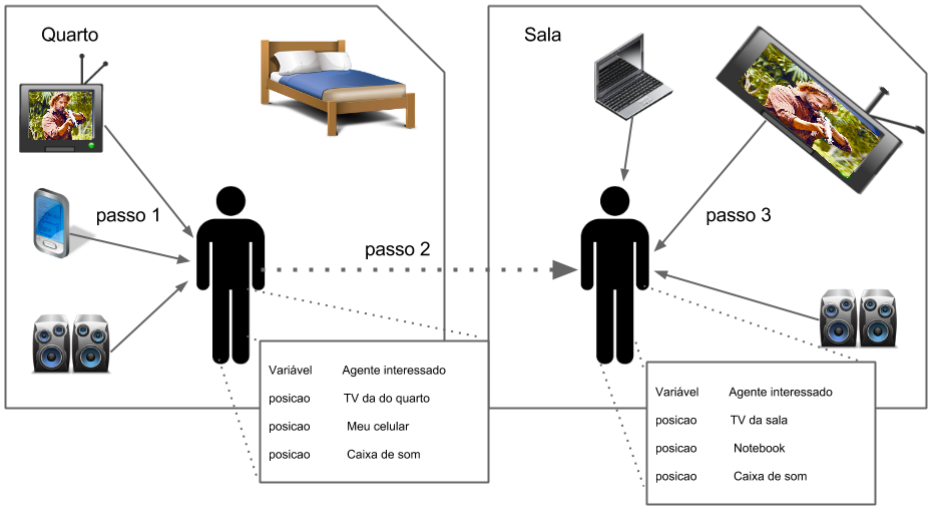

Figura 20 - Aplicação de Exibição ou Audição de Conteúdo em Dispositivos Móveis, Televisões e Rádios

Esta aplicação utilizando a implementação original da SmartAndroid ou utilizando a infraestrutura do protocolo REPI terá diversos registros de interesse como mostrado na Figura 21. Neste caso, 4 Agentes de Recursos estão interessados na variável de contexto posição do agente “andre.ra”, porém em um ambiente ubíquo real, este número poderá ser muito maior.

```
InterestAPI ia = InterestAPIImpl.getInstance();
try {
    String interest = "andre.ra/myPosition";
    ia.registerInterest(smartphoneRans, interest, callback);
    ia.registerInterest(tabletRans, interest, callback);
    ia.registerInterest(computerRans, interest, callback);
    ia.registerInterest(tvRans, interest, callback);
} catch (Exception e) {
    System.out.println(e);
}
```

Figura 21 - Registro de interesse de diversos Agentes de Recurso na posição do agente "andre.ra"

Quando ocorrer uma alteração na variável de contexto posição do agente “andre.ra”, a implementação original da SmartAndroid criará um Stub para cada um dos agentes interessados nesta posição (veja a Figura 22). A criação de um Stub envolve o estabelecimento de uma nova conexão, construção dos objetos necessários e identificação do RANS para o qual a mensagem será enviada.

```
public void notifyStakeholders(String contextVariable, Object value) {
    for (Stakeholder stakeholder : stakeholders) {
        if (stakeholder.getContextVariable().equals(contextVariable) ||
            stakeholder.getContextVariable().equalsIgnoreCase("all")) {

            new ResourceAgentStub(stakeholder.getRANS()).notificationHandler(myRANS, contextVariable, value);
            Log.d("SmartAndroid", String.format("notifying stakeholder: %s " +
                    "contextVariable: %s value: %s", stakeholder.getRANS(),
                    contextVariable, value));
        }
    }
}
```

Figura 22 - Notificação dos Stakeholders interessados em uma variável de contexto utilizando a implementação sem o protocolo REPI

Apesar de essas estruturas poderem ser cacheadas, no cenário onde o conjunto de interessados muda dinamicamente, a utilização do protocolo REPI traz a possibilidade de economia de recursos enviando uma única mensagem para todos os interessados na variável posição.

Veja na Figura 23 como fica o método “notifyStakeholders”, utilizando o protocolo REPI. Desta maneira, não é necessário percorrer os 4 agentes interessados, e constantes mudanças na variável posição serão de processamento mais rápido. Essa abordagem é particularmente útil para os cenários onde a troca de dados pode ser intensa. Na Seção 4.5.2 é discutido como o REPI evita que a rede seja inundada.

```
public void notifyStakeholdersNew(String contextVariable, Object value) {
    InterestAPI ia = InterestAPIImpl.getInstance();
    try {
        String interest = myRANS + "/"+ contextVariable;
        ia.sendMessage(interest, (String) value);
    } catch (Exception e) {
        System.out.println(e);
    }
}
```

Figura 23 - Notificação dos Stakeholders interessados em uma variável de contexto utilizando a implementação com o protocole REPI

Levando em consideração requisitos de economia de energia, a redução no número de mensagens enviadas no momento de notificação dos interessados pode representar ganhos na redução do consumo de energia dos dispositivos.

Em relação à etapa (iv), imagina-se um cenário de uma casa inteligente composta por uma pessoa assistindo uma série de televisão no quarto. Neste cenário, a pessoa resolve ir para a sala, pois a televisão é maior. A pessoa então sai do quarto e caminha até a sala. Quando a pessoa saiu do quarto, a aplicação de exibição de conteúdo interrompeu a exibição da série, e todos os ARs presentes no quarto que estavam interessados na posição do AR Pessoa foram notificados que o AR Pessoa saiu do quarto. Ao identificar que a pessoa chegou à sala, os ARs da sala que podem exibir conteúdo registram interesse na posição da pessoa e oferecem a possibilidade de dar continuidade à série de televisão que estava sendo assistida. Caso a pessoa volte a se movimentar, todos os ARs registrados no interesse posição serão notificados. Neste caso, a aplicação utilizado o protocolo REPI seria capaz de enviar uma única mensagem para a rede através do interesse "andre.ra/myPosition” fazendo com que a mensagem chegasse a todos os dispositivos interessados na posição do AR Pessoa.

6.2.1.3 APLICAÇÃO DE NOTIFICAÇÃO DE INCÊNDIO, ARROMBAMENTO OU ASSALTO

Esta Seção aborda a questão 3, relacionada ao agente ser capaz de lidar quando é preciso enviar uma mensagem a um grande número de agentes interessados. Essa questão é explorada através do desenvolvimento de uma aplicação de notificação de incêndio, arrombamento ou assalto.

Esta aplicação é composta de diversos sensores responsáveis por identificar se um incêndio foi iniciado, se houve um arrombamento ou uma tentativa de assalto e de um conjunto de dispositivos capazes de notificar a ocorrência do fato, como por exemplo, um alarme, uma notificação visual ou uma ocorrência para os bombeiros.

Cumprindo a etapa (ii), a questão 3, em termos do Framework UFF, pode ser representada através de um AR responsável por identificar a ocorrência de um incêndio e de um conjunto de ARs capazes de disparar um alarme ou de destacar o ocorrido, como, por exemplo, alarmes sonoros, celulares, televisões, entre outros. Quando ocorre um incêndio, o AR responsável pelo monitoramento precisa notificar todos os dispositivos desse tipo que está ocorrendo um incêndio. Para isso, é disparada uma mensagem para cada um dos ARs interessados na variável de contexto "incêndio" do AR responsável pelo monitoramento. A Figura 24 demonstra essa interação.

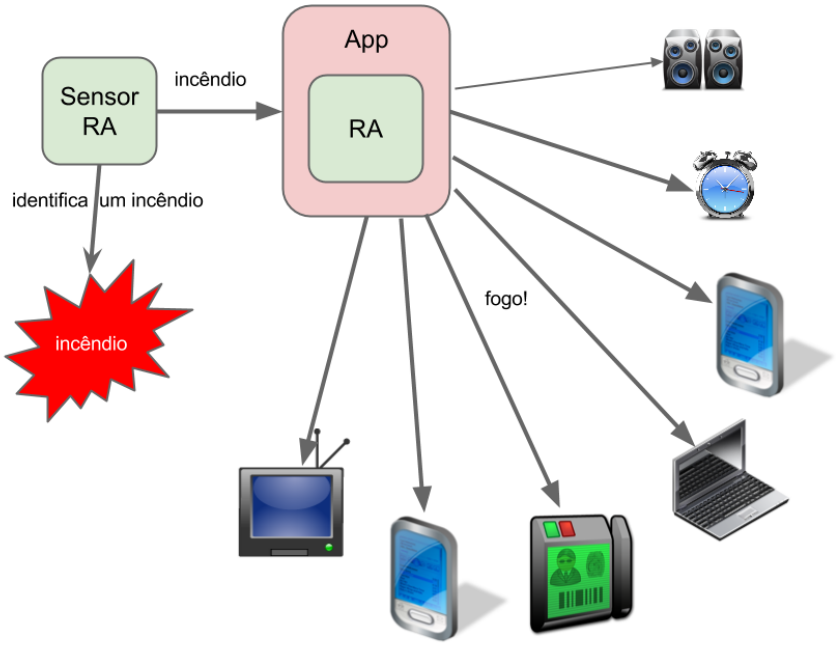

Figura 24 - Representação da questão de competência 3

Cumprindo a etapa (iii), a aplicação de notificação de incêndio, arrombamento ou assalto é modelada na Figura 25. A Figura 25 descreve os principais componentes envolvidos, sendo eles: o AR responsável pelo monitoramento de incêndio e diversos dispositivos presentes na casa capazes de disparar um alarme ou exibir um aviso de incêndio. Inicialmente o AR responsável pelo monitoramento recebe o registro de interesse de diversos dispositivos capazes de disparar um alarme ou exibir um aviso de incêndio (passo 1). Ao ocorrer um incêndio, este AR identifica a ocorrência (passo 2). Após identificar o incêndio, este mesmo agente notifica todos os ARs interessados na variável de contexto "incêndio" (passo 3).

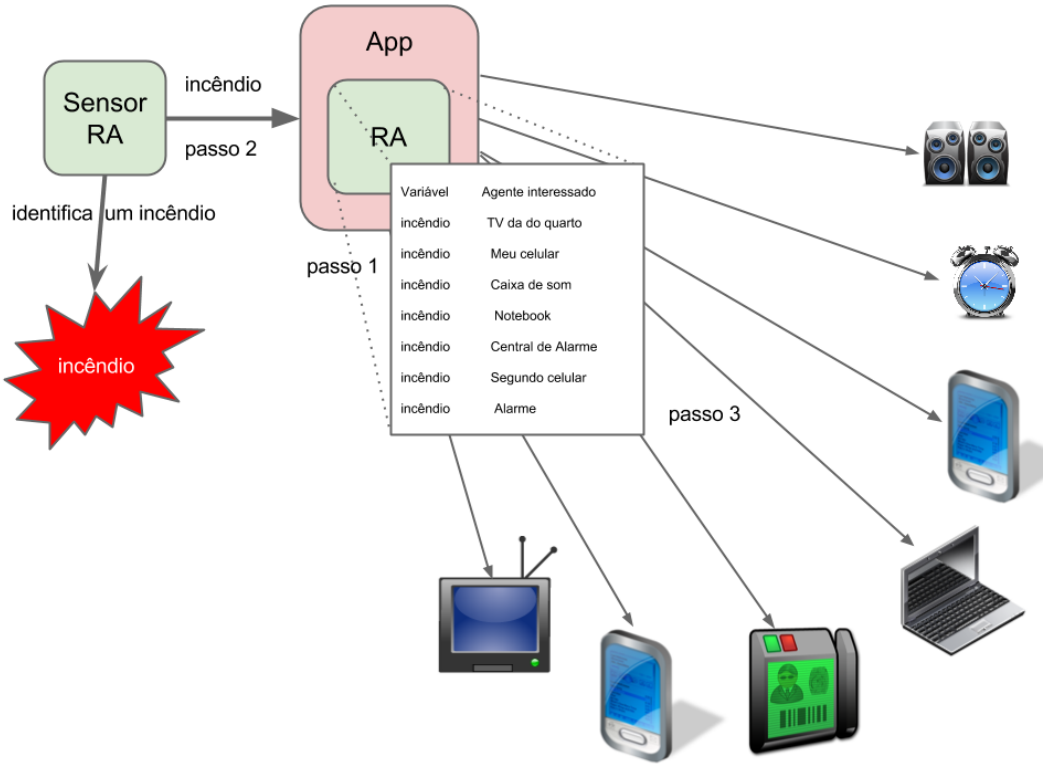

Figura 25 - Aplicação de Notificação de Incêndo, Arrombamento e Assalto

Esta aplicação se beneficia diretamente da utilização do protocolo REPI, uma vez que seria possível utilizar o mecanismo de propagação da mensagem REPI, a fim de se cobrirem todos os Agentes de Recurso interessados rapidamente e com o mínimo gasto. Veja na Figura 26 como fica o disparo de um aviso de incêndio, utilizando o protocolo REPI.

```
InterestAPI ia = InterestAPIImpl.getInstance();
try {
    String interest = "dangerMonitoringApp://alarm";
    ia.sendMessage(interest, "fire");
} catch (Exception e) {
    System.out.println(e);
}
```

Figura 26 - Aviso de incêndio para os agentes responsáveis por alarmar

Na implementação da SmartAndroid, se o agente responsável por informar o alarme morrer antes da notificação de todos os agentes interessados somente alguns agentes estarão cientes do perigo. No caso da utilização do protocolo REPI, uma vez enviada para a rede, o próprio funcionamento do protocolo faz com que a mensagem seja encaminhada colaborativamente para todos os interessados (veja as seções 4.3 e 4.5.1). Entretanto, é relevante considerar as questões relativas a taxa de entrega abordadas na Seção 4.5.4.

Em relação à etapa (iv), imagina-se um cenário de uma casa inteligente composta por um sensor capaz de identificar um incêndio e diversos dispositivos capazes de disparar um alarme sonoro ou visual da ocorrência de um incêndio na casa. Neste cenário, ao identificar um incêndio, todos os dispositivos interessados na variável de contexto "incêndio" seriam notificados. Ao receber essa notificação, cada um dos dispositivos seria responsável por disparar o alarme mais adequado de acordo com a sua característica. Por exemplo, alarmes emitiriam um alerta sonoro, enquanto uma televisão emitiria um alerta visual.

6.2.1.4 APLICAÇÃO DE TROCA DE CONTEÚDO ENTRE ADERENTES A UM PADRÃO

Esta Seção aborda a questão 4, relacionada ao agente ser capaz de criar uma rede sobreposta entre agentes com interesses em comum. Essa questão é explorada através do desenvolvimento de uma aplicação de troca de conteúdo entre dispositivos aderentes a um padrão.

Esta aplicação visa trocar conteúdo como textos, vídeos e músicas entre aparelhos aderentes a um padrão. ARs que estão em um mesmo ambiente tendem a trocar informações comuns entre si. Dispositivos podem possuir interesses comuns para troca de informações, como dispositivos de um mesmo fabricante, de um mesmo propósito ou que adotam um padrão comum de interação.

Cumprindo a etapa (ii), a questão 4, em termos do Framework UFF, pode ser representada através de um conjunto de ARs que adotam um padrão comum de interação e que por isso se registram em interesses comuns. A Figura 27 demonstra grupos de agentes que possuem interesses em comum e que assim criam redes sobrepostas para comunicação entre eles.

Figura 27 - Representação da questão de competência 4

Cumprindo a etapa (iii), a aplicação de troca de conteúdo entre dispositivos aderentes a um padrão é modelada na Figura 28. A Figura 28 descreve os principais componentes envolvidos, sendo eles: dois grupos de agentes que criam entre si redes sobrepostas para a troca informações comuns entre eles. Nela é possível observar que os ARs envolvidos possuem os mesmos interesses e os utilizam para a troca de informações relevantes.

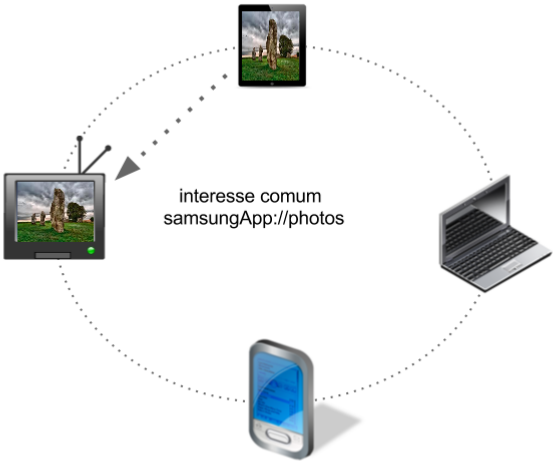

Figura 28 - Aplicação de Troca de Conteúdo entre dispositivos Aderentes a um Padrão

A abordagem atual de comunicação da SmartAndroid não fornece mecanismos nativos para a criação de redes sobrepostas de forma imediata. Para isso, o projetista de aplicações precisaria criar o suporte a nível de aplicação, de forma a se ter a troca de mensagens entre diversas redes entre os agentes. Uma outra opção seria utilizar uma plataforma específica para este fim, porém acarretaria em mais um esforço para o programador. Neste caso, esse suporte poderia ser embutido no framework, facilitando assim o seu uso.

Já o funcionamento do protocolo REPI permite, através do registro de interesses comuns ou padronizados, o estabelecimento de redes sobrepostas entre os agentes. Desta forma, é possível ter diversas redes com um canal comum de comunicação, conectando agentes, aplicações e dispositivos com um interesse em comum. Na Figura 29 é possível ver uma sequência de registros de interesses de AR que seguem um padrão. Vale destacar que, utilizando o protocolo REPI, a API de comunicação é a mesma, diminuindo o esforço do programador.

A primeira sequencia de registros de interesses da Figura 29 cria uma rede sobreposta para troca e controle de eventos dos ARs Smartphone, Tablet, Notebook e TV através do interesse “samgungApps://eventControlApp://eventBus”, enquanto a segunda sequência de registros de interesses cria uma rede sobreposta para troca

de conteúdo entre esses mesmos ARs através do interesse “samgungApps:://contentExchangeApp://contentBus.

Destaca-se neste ponto a possibilidade de se criarem padrões, regras ou boas práticas para a definição de interesses comuns, de forma que seja possível criar um protocolo único de troca de informações não somente entre dispositivos de um mesmo fabricante ou que seguem um padrão, mas também de aplicações ubíquas construídas sob um mesmo domínio.

Em relação à etapa (iv), imagina-se um cenário de uma casa inteligente composta por um conjunto de dispositivos seguindo um padrão de comunicação para troca de conteúdo como, por exemplo, um Tablet, um Notebook, uma TV e um Smartphone. Neste cenário, caso uma pessoa desejasse transferir uma foto do tablet para a TV, como ambos estabelecem um padrão de comunicação, a pessoa poderia transferir a foto entre qualquer um dos dispositivos. Veja na Figura 28 um exemplo de transmissão de foto.

```
InterestAPI ia = InterestAPIImpl.getInstance();
try {
    String interestEvents = "samsungApps://eventControlApp://eventBus";
    ia.registerInterest(smartphoneSamsungRans, interestEvents, callback);
    ia.registerInterest(tabletSamsungRans, interestEvents, callback);
    ia.registerInterest(notebookSamsungRans, interestEvents, callback);
    ia.registerInterest(tvSamsungRans, interestEvents, callback);

    String interestContent = "samsungApps://contentExchangeApp://contentBus";
    ia.registerInterest(smartphoneSamsungRans, interestContent, callback);
    ia.registerInterest(tabletSamsungRans, interestContent, callback);
    ia.registerInterest(notebookSamsungRans, interestContent, callback);
    ia.registerInterest(tvSamsungRans, interestContent, callback);
} catch (Exception e) {
    System.out.println(e);
}
```

Figura 29 - Registro de interesse de duas aplicações distintas que adotam um padrão comum de interação e que utilizam a criação de interesses como forma de construção de uma rede sobreposta entre dispositivos

6.3 AVALIAÇÃO QUANTITATIVA

Além da avaliação baseada nas questões de competências descritas na Seção anterior, alguns experimentos foram executados, visando identificar como o protocolo REPI se comporta em um cenário típico de um ambiente ubíquo. Embora

[MORAES, H. F. et al., 2012a] já tenham feito diversas análises quanto à taxa de entrega das mensagens, obtendo uma taxa de entrega acima de 96% (veja o Apêndice A3), a avaliação a seguir foi implementada em um ambiente doméstico real, utilizando tanto a implementação atual da SmartAndroid quanto a implementação com o protocolo REPI integrado.

Para esse experimento foram utilizados dispositivos Android (versão 2.3) instalados com o *Daemon* REPI, uma aplicação de geração, recebimento e contabilização de mensagens e uma rede local wifi para a conexão entre os dispositivos. Foram executadas 10 baterias para cada uma das implementações, sendo distribuídas em envios de 10.000 e 20.000 mensagens. Antes de cada execução foi garantido que somente os serviços essenciais para o funcionamento do dispositivo estavam em execução. Além disso, os dispositivos não possuíam chip telefônico para prevenir o acesso a rede de telefonia durante os testes.

Diante disso, os testes efetuados tomaram como critério de avaliação pontos relevantes discutidos ao longo do trabalho: a troca de mensagens (abordada na Seção 4.5.2), o gasto de energia (abordada na Seção 4.5.6) e a taxa de entrega das mensagens (abordada na Seção **Error! Reference source not found.**). O cenário preveem um AR enviando diversas mensagens para outro AR presente em um outro dispositivo, variando a quantidade de mensagens e se o parâmetro que permite configurar o encaminhamento de mensagens (abordado no Apêndice A2 e disponível na implementação integrada ao REPI) estava habilitado ou não. Este parâmetro de configuração do encaminhamento de mensagens é potencialmente útil em uma rede wifi, dado que os dispositivos já estão interligados.

Visando analisar as mensagens recebidas pelo Daemon na implementação integrada ao REPI, foi feita uma tentativa de utilização de uma API do Daemon capaz de fornecer dados estatísticos das mensagens que passaram por ele. Entretanto, esta ainda é uma API em desenvolvimento, e, em de diversas tentativas, alguns problemas foram observados na utilização da mesma. Para contornar essa limitação, os logs gerados pelo Daemon foram utilizados. Os dados que a API forneceria foram expostos no Daemon, o que possibilitou a coleta dos dados após as interações.

Já na implementação atual da SmartAndroid, os logs de envio e recebimento das mensagens foram suficientes para a análise das entregas das mensagens.

A Tabela 2 e a Tabela 3 abaixo expõem os resultados médios obtidos na implementação integrada ao protocolo REPI. O HTL (Hop-to-live) é a quantidade de saltos entre dispositivos necessários para a mensagem ser entregue ao destino.

Envio de 10.000 mensagens	taxa de entrega	% de queda da bateria	Mensagens recebidas	HTL médio
Com o encaminhamento das mensagens	98,46%	inferior a 1%	11.434	1
Sem o encaminhamento	99,42%	inferior a 1%	9.932	1

Tabela 2 - Envio de 10.000 mensagens

Envio de 20.000 mensagens	taxa de entrega	% de queda da bateria	Mensagens recebidas	HTL médio
Com o encaminhamento das mensagens	99,47%	inferior a 1%	20.954	1
Sem o encaminhamento	99,53%	inferior a 1%	19.910	1

Tabela 3 - Envio de 20.000 mensagens

Com os resultados acima, pode-se observar que o percentual de queda da bateria foi mínimo e que, conforme já discutido anteriormente, de fato a taxa de entrega das mensagens fica próxima a 100%, porém não garante a entrega por completo com taxas de perda abaixo de 1%.

No caso do envio das mensagens com o encaminhamento, o número de mensagens recebidas foi ligeiramente maior que o número de mensagens enviadas. Já no caso do envio das mensagens sem o encaminhamento, o número de mensagens recebidas foi bem próximo ao número de mensagens enviadas, não sendo igual pela perda de mensagens. Já o HTL médio foi 1, já que os dispositivos estão interligados.

Além disso, a habilitação do parâmetro que define se as mensagens serão encaminhadas (abordado no Apêndice A2) possibilitou a redução no número de mensagens recebidas e, consequentemente, processadas. Isso é importante pois previne o gasto desnecessário de energia no processamento de mensagens e na disseminação de mensagens já recebidas na rede.

A Tabela 4 e a Tabela 5 abaixo expõem os resultados médios obtidos na implementação atual da SmartAndroid. Em ambas as Tabelas é possível observar que a taxa de entrega foi de 100% nos experimentos realizados e que o número de mensagens recebidas foi igual ao número de mensagens enviadas. Além disso, assim como a implementação integrada ao protocolo REPI a queda da bateria foi inferior a 1%.

	taxa de entrega	% de queda da bateria	Mensagens recebidas
Envio de 10.000 mensagens	100%	inferior a 1%	10.000

Tabela 4 - Envio de 10.000 mensagens na implementação atual da SmartAndroid

	taxa de entrega	% de queda da bateria	Mensagens recebidas
Envio de 20.000 mensagens	100%	inferior a 1%	20.000

Tabela 5 - Envio de 20.000 mensagens na implementação atual da SmartAndroid

Diante dos resultados acima, observa-se que os resultados obtidos em ambas as implementações são semelhantes, diferindo na taxa de entrega e no número de mensagens recebidas. Comparando a implementação atual com a implementação integrada ao REPI pode-se observar que as taxas de entrega da implementação integrada ao REPI foram sempre inferiores 100%, enquanto na implementação atual 100% das mensagens foram entregues em todos os experimentos. Na implementação integrada ao REPI com o encaminhamento das mensagens, o número de mensagens recebidas é superior ao número de mensagens enviadas.

Isto se dá pela propagação das mensagens intrínseco ao protocolo. Já na implementação atual, o número de mensagens enviadas é igual ao número de mensagens recebidas.

6.4 CONCLUSÃO DA AVALIAÇÃO

O resultado obtido com a construção da Aplicação de Monitoramento de Acidentes com Idosos (AMAI), apresentada na Seção 6.2.1.1, demonstra que o código da solução é o mesmo em ambas as abordagens. Isto mostra que API de comunicação mantém uma interface mínima e única com as consultas comuns independente do protocolo usado na comunicação. Destaca-se, porém, que o protocolo REPI se mostra resiliente no caso de uma falha no Repositório de Recursos sob a ótica da necessidade de uma busca não poder ser feita no Repositório de Recursos (veja a Seção 6.2.1.1). Ainda assim, um maior suporte a tratamento de erros para suprir as questões levantadas na Seção **Error! Reference source not found.** é relevante.

A Aplicação de Exibição ou Audição de Conteúdo em Dispositivos Móveis, Televisões e Rádios (AEACDM), apresentada na Seção 6.2.1.2, demonstra que, para um cenário onde diversos agentes estão interessados em um único agente, a abordagem utilizando o protocolo REPI pode ser bem interessante. Isto ocorre por alguns fatores como: código de desenvolvimento mais simples e menor consumo de energia devido a redução no número de mensagens enviadas. Para o caso de uma troca de dados intensa, a simplificação de enviar a mensagem para a rede ao invés de mensagens individuais pode ser particularmente útil. Ainda assim, é importante considerar os critérios expostos na Seção 4.5.2, que demonstram como o REPI lida com a sobrecarga de mensagens na rede.

Entretanto, é importante frisar que o protocolo se destaca quando de fato existem muitos agentes interessados em um mesmo AR. Nos casos em que apenas uma quantidade razoável de agentes estão interessados em um mesmo agente, a abordagem atual, sem REPI, é capaz de atender perfeitamente os requisitos sem ter uma perda significativa na latência e consumo de energia. Além disso, a implementação atual poderia ser reconstruída, com um esforço considerável, utilizando endereços Multicast, para disseminação da mensagem de acordo com os interesses registrados.

A Aplicação de Controle de Incêndio, Arrombamento ou Assalto (ACIAA), apresentada na Seção 6.2.1.3, mostra que, para este cenário analisado, o código gerado pela abordagem via protocolo REPI é mais simples, pois só é necessário o disparo de uma única mensagem a partir do agente que identifica o incêndio, arrombamento etc.. Para um ambiente com poucos agentes, o impacto gerado pela necessidade de identificar previamente os agentes que precisam ser notificados na abordagem atual pode não ser um problema. Porém, um aspecto importante é o cenário onde o agente responsável por notificar morre antes que todos os agentes interessados estejam notificados. Na abordagem via REPI, este problema não ocorre visto que a mensagem é disparada uma única vez, e, de maneira colaborativa o protocolo REPI faz com que as mensagens sejam entregues aos respectivos interessados.

A Aplicação de Troca de Conteúdo entre Dispositivos Aderentes a um Padrão (ATCDAP), apresentada na Seção 6.2.1.4, constata que, para o cenário de criação de redes sobrepostas, a utilização da abordagem atual fará com que seja necessário por parte do projetista desenvolver todo o suporte necessário. Isso acarretará em mais desenvolvimento e trabalho por parte do usuário do framework. Em contrapartida, a abordagem via REPI já provê todo o suporte para criação de redes sobrepostas, utilizando-se da modelagem via interesses nativa do protocolo. É possível notar que a escolha adequada dos interesses pelo projetista da aplicação permite agrupamentos e o estabelecimento de relações entre os Agentes de Recurso. Entretanto, uma análise mais profunda quanto a criação adequada de interesses possibilitaria uma maior manutenabilidade e capacidade de evolução do sistema.

Observa-se que a criação de diversas aplicações possibilitou a validação da implementação original da SmartAndroid e da implementação com protocolo REPI. Para a avaliação foram levados em consideração cenários reais que avaliam questões de competências relevantes para o desenvolvimento de sistemas ubíquos inteligentes. A utilização de uma API única de comunicação facilitou o desenvolvimento das aplicações, bem como ajudou a extrair em termos de essência, o que a API de comunicação deveria ter de operações.

A análise acima mostra que a implementação atual da SmartAndroid atende à maioria dos cenários, não sendo um impeditivo para a construção de aplicações

ubíquas relativamente pequenas; porém alguns cenários não puderam ser inteiramente cobertos pela API atual ou se mostraram menos eficientes.

É importante destacar que o protocolo REPI tem o potencial para atender requisitos de grandes aplicações ubíquas, com centenas ou milhares de ARs. Em particular, tem características que o fazem adequado a aplicações móveis, realizando interações de forma dinâmica.

Diante disso, é possível mostrar que o uso do protocolo REPI fornece mais um conjunto de ferramentas importantes para o desenvolvimento de aplicações ubíquas. Mesmo que, com certas limitações, características e considerações (veja Apêndice A1) este protocolo poderia ser utilizado para os mecanismos de comunicação ou como complemento da abordagem atual, suprindo e aperfeiçoando os fluxos e cenários onde ela não se mostra muito adequada.

CAPÍTULO 7 CONCLUSÃO

7.1 CONTRIBUIÇÕES

Como contribuição desta dissertação, destaca-se a definição e implementação de uma API única de comunicação que integra tanto o procedimento de chamada remota (RPC), quanto a comunicação por interesses (através do paradigma publish-subscribe). Esta API permite o desenvolvimento de aplicações distribuídas baseadas em ARs (ou objetos distribuídos). Além disso, a API foi concebida de forma que possa ser usada transparentemente em aplicações SmartAndroid ou REPI.

Apresenta-se, também, uma comparação entre duas propostas que possibilitam o desenvolvimento de aplicações ubíquas e inteligentes cuja comunicação entre os objetos é expressa através de interesses.

O desenvolvimento de um conjunto de aplicações é descrito, utilizando a infraestrutura da SmartAndroid e a API de comunicação orientada a interesses. Tais aplicações servem como referência para a construção de novas aplicações que usem a infraestrutura da SmartAndroid ou o protocolo REPI.

Demonstra-se, ainda, o uso do protocolo REPI em um ambiente ainda não explorado, trazendo a possibilidade de um novo campo de pesquisa para o protocolo.

7.2 LIMITAÇÕES

A API de desenvolvimento do protocolo REPI ainda não possui mecanismos para a garantia de entrega das mensagens, conforme já discutido em 4.5.4. Embora possua uma taxa de entrega de mensagens acima de 96%, algumas aplicações necessitam que 100% das mensagens sejam entregues.

As aplicações foram desenvolvidas utilizando a plataforma Android (versões 2.3 e 3.2), porém a API de desenvolvimento do protocolo REPI não suporta a versão 4 da plataforma Android. Vale destacar que o Daemon responsável pelo envio das mensagens REPI não possui versões para o sistemas iOS, Windows Phone e outros ambientes de comunicação sem fio. Entretanto, não existem dificuldades conceituais para a implementação da API e da proposta nestas e em outras plataformas, desde que sejam abertas.

A utilização do modo de funcionamento Ad-hoc acarreta em grande gasto de energia por parte do *Daemon* da API conforme destacado na Seção 4.5.6.

7.3 TRABALHOS FUTUROS

Como trabalhos futuros, é importante destacar que o desenvolvimento de aplicações reais permitiria uma avaliação mais completa das propostas associadas a este trabalho. Além disso, essas aplicações ajudariam na identificação de cenários não mapeados.

A avaliação quantitativa realizada permitiu comparar minimamente ambas as abordagens, porém com o desenvolvimento de aplicações reais seria possível efetuar uma análise mais completa da entrega das mensagens e do número de mensagens trafegadas. Nesse cenário, seria importante avaliar como a entrada e saída de novos ARs reflete nas mensagens trafegadas e no recebimento de mensagens.

O mapeamento entre o sistema de nomes de agentes RANS e a disponibilização dos agentes na internet através de IP e DNS permitiria o protocolo ser escalado para uma quantidade maior de nós (veja Seção 4.5.8). Esse mapeamento poderia ser criado através de mecanismos semelhantes à técnica *Network Address Translation* (NAT) onde seria possível um Agente de Recurso possuir um IP externo que o localiza. Já no caso dos dispositivos operarem no nível da chamada *Internet of things*, este processo torna-se mais simples, uma vez que cada dispositivo poderia possuir um IP particular possibilitado pelo IPV6.

O uso do protocolo REPI para o desenvolvimento de aplicações ubíquas abre uma área de pesquisa na definição de melhores práticas para a criação de interesses focados no domínio de aplicações ubíquas inteligentes. Um modelo formal de ontologia poderia ser criado para a padronização dos interesses criados neste domínio.

Apesar da possibilidade da troca de informações poder estar atrelada a uma rede segura Wi-fi, aspectos relacionados à segurança no tráfego de mensagens internamente à rede via protocolo REPI, além dos já discutidos na Seção 4.5.5, precisam ser aprofundados.

Com o desenvolvimento de novas aplicações via SmartAndroid, faz-se necessário estabelecer uma loja única de aplicativos e exemplos de aplicações. Esta loja facilitaria tanto o usuário final que usa os aplicativos, como o desenvolvedor que

os desenvolve. Além disso, seria possível garantir a confiabilidade dos aplicativos, pois preveniria que softwares mal intencionados fossem colocados no ambiente inteligente.

REFERÊNCIAS

AUGUSTO, J. C.; MCCULLAGH, P. **Ambient Intelligence: Concepts and Applications.** Computer Science and Information Systems, v. 4, n. 1, p. 1 – 27, 2007.

ARMAC, I.; RETKOWITZ, D. **Simulation of Smart Environments**. IEEE International Conference on Pervasive Services. p.322-331, 2007. IEEE.

BALDAUF, M.; DUSTDAR, S.; ROSENBERG, F. **A survey on context-aware systems**. International Journal of Ad Hoc and Ubiquitous Computing, v. 2, n. 4, p. 263, 2007. Inderscience Publishers.

BARRETO, D.; ERTHAL, M.; MARELI, D.; LOQUES, O. **Uma Interface de Prototipagem para Aplicações Pervasivas**. XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC). p.629–642, 2013. SBRC.

CALLON, R. **Use of OSI IS-IS for Routing in TCP/IP and Dual Environments.** ,1990. Disponível em: <http://www.ietf.org/rfc/rfc1195.txt>. Acesso em: 20/2/2013.

CARVALHO, S. T.; ERTHAL, M.; MARELI, D.; **Monitoramento Remoto de Pacientes em Ambiente Domiciliar**. XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos-Salao de Ferramentas, Gramado, RS, Brasil. p.1005-1012, 2010. SBRC.

CLARK, D. **The design philosophy of the DARPA internet protocols.** ACM SIGCOMM Computer Communication Review, v. 18, n. 4, p. 106-114, 1988. New York, NY, USA: ACM Press.

DEY, A.; ABOWD, G.; SALBER, D. **A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications.** Human-Computer Interaction, v. 16, n. 2, p. 97-166, 2001. L. Erlbaum Associates Inc.

DUTRA, R. DE C.; GRANJA, R. S.; MORAES, H. F.; AMORIM, C. L. **REPI: Rede de comunicação Endereçada Por Interesses.** XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. p.99-112, 2010. SBRC.

DUTRA, R. C.; MORAES, H. F.; AMORIM, C. L. **Interest-Centric Mobile Ad Hoc Networks**. 2012 IEEE 11th International Symposium on Network Computing and Applications. p.130–138, 2012. IEEE.

EUGSTER, P. T.; FELBER, P. A.; GUERRAOUI, R.; KERMARREC, A.-M. **The many faces of publish/subscribe**. ACM Computing Surveys, v. 35, n. 2, p. 114-131, 2003.

FILLINGER, A.; HAMCHI, I.; DEGRE, S. et al. **Middleware and Metrology for the Pervasive Future**. IEEE Pervasive Computing, v. 8, n. 3, p. 74-83, 2009. IEEE.

FORD, B.; SRISURESH, P.; KEGEL, D. **Peer-to-peer communication across network address translators**. ATEC '05 Proceedings of the annual conference on USENIX Annual Technical, p. 13, 2005. USENIX Association.

GRÜNINGER, M.; FOX S. **Methodology for the Design and Evaluation of Ontologies**. International Joint Conference on Artificial Inteligence (IJCAI95), Workshop on Basic Ontological Issues in Knowledge Sharing. 1995.

HELAL, S.; MANN, W.; EL-ZABADANI, H.; et al. **The Gator Tech Smart House: a programmable pervasive space.** Computer, v. 38, n. 3, p. 50–60, 2005. IEEE.

HOLDREGE, M.; SRISURESH, P. **Protocol Complications with the IP Network Address Translator**. RFC 3027, Internet Engineering Task Force, 2001. Disponível em: <http://www.ietf.org/rfc/rfc3027.txt>. Acesso em: 08/12/2013.

JACOBSON, V.; SMETTERS, D. K.; THORNTON, J. D. et al. **Networking named content.** Proceedings of the 5th international conference on Emerging networking experiments and technologies - CoNEXT '09. p.1, 2009. New York, NY, USA: ACM Press.

KIM, I.; PARK, H.; NOH, B. et al. **Design and Implementation of Context-Awareness Simulation Toolkit for Context learning**. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing - Vol 2 - Workshops. v. 2, p.96-103, 2006. IEEE.

KOPONEN, T.; CHAWLA, M.; CHUN, B.-G. et al. **A data-oriented (and beyond) network architecture.** ACM SIGCOMM Computer Communication Review, v. 37, n. 4, p. 181, 2007. New York, NY, USA: ACM Press.

MARELI, D. Um **Framework de Desenvolvimento de Aplicações Ubíquas em Ambientes Inteligentes**. [Dissertação]. Niterói: Universidade Federal Fluminense, Instituto de Computação, 2013

MARELI, D.; ERTHAL, M.; BARRETO, D.; LOQUES, O. **Um Framework de Desenvolvimento de Aplicações Ubíquas em Ambientes Inteligentes**. XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC). p.643–656, 2013. SBRC.

MORAES, H. F.; BENITEZ, N. R. P.; DUTRA, R. C.; AMORIM, C. L. **On developing interest-centric applications for ad hoc networks**. 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM). p.1–3, 2012a. IEEE.

MORAES, H. F.; DUTRA, R. DE C.; AMORIM, C. L. **REPI : Um Protocolo Peer-to-Peer para Aplicações Orientadas a Interesses na Internet.** XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. p.60-73, 2012b. SBRC.

PARK, J.; MOON, M.; HWANG, S.; YEOM, K. **CASS: A Context-Aware Simulation System for Smart Home.** 5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007). p.461-467, 2007. IEEE.

RANGANATHAN, A.; CAMPBELL, R. H. **A middleware for context-aware agents in ubiquitous computing environments**. ACM/IFIP/USENIX 4th International Middleware Conference. p. 143–161, 2003. Springer-Verlag New York, Inc.

RANGANATHAN, A.; CHETAN, S.; AL-MUHTADI, J.; CAMPBELL, R. H.; MICKUNAS, M. D. **Olympus: A High-Level Programming Model for Pervasive Computing Environments**. Third IEEE International Conference on Pervasive Computing and Communications. p.7-16, 2005. IEEE.

SOCOLOFSKY, T.; KALE, C. **A TCP/IP Tutorial.** ,1991. Disponível em: <http://www.ietf.org/rfc/rfc1180.txt>. Acesso em: 08/12/2013.

VAN NGUYEN, T.; KIM, J. G.; CHOI, D. **ISS: the interactive smart home simulator.** The 11th International Conference on Advanced Communication Technology, p. 1828-1833, 2009. IEEE.

WEISER, M. **The computer for the 21st century.** Scientific American, v. 265, n. 3, p. 94–104, 1991. New York, NY, USA: ACM Press.

WEISER, M. **Some computer science issues in ubiquitous computing**. Communications of the ACM, v. 36, n. 7, p. 75–84, 1993. New York, NY, USA: ACM Press.

WEISER, M. **Creating the invisible interface.** Proceedings of the 7th annual ACM symposium on User interface software and technology - UIST ’94. p.1, 1994. New York, NY, USA: ACM Press.

WEISER, M. **The world is not a desktop.** Interactions, v. 1, n. 1, p. 7–8, 1994. New York, NY, USA: ACM Press.

WELLNER, P.; MACKAY, W.; GOLD, R. **Computer-Augmented Environments: Back to the real world**. Communications of the ACM, v. 36, n. 7, p. 24–27, 1993. New York, NY, USA: ACM Press.

APÊNDICE A – CONSIDERAÇÕES SOBRE A ADOÇÃO DO PROTOCOLO REPI EM UM AMBIENTE UBÍQUO E INTELIGENTE

Um ambiente ubíquo e inteligente, principalmente em um cenário de uma *smarthome,* prevê tipicamente uma quantidade finita de dispositivos. Para se avaliar corretamente qualquer nova proposta ou abordagem, é preciso ter em mente o cenário real de adoção.

Impulsionada pelo crescente avanço da Internet e pela conectividade que ela proporciona, bem como pelo constante crescimento de uso de dispositivos móveis, tem sido cada vez mais comum a adoção de redes wifi em ambientes domésticos. Essas redes integram os dispositivos presentes na casa, facilitam a troca de informações e conteúdo entre eles e fornecem um protocolo padrão para conectividade em um ambiente doméstico.

Diante dos pontos acima é fundamental efetuar uma análise das características e limitações do protocolo perante um ambiente real de uso.

A1 INSTALAÇÃO DO DAEMON REQUER PERMISSÃO DE SUPERUSUÁRIO

O processo de instalação descrito na Seção 4.5.3 estabelece a necessidade de possuir um dispositivo com permissão de superusuário. Essa limitação se dá pelo proprietário da tecnologia onde a API está inserida, porém pode ser um fator crucial na adoção do protocolo. Embora haja diversos tutoriais, vídeos explicativos e manuais de como se obter permissão de superusuário, esse processo é arriscado, exige um certo grau de entendimento de tecnologia, e o dispositivo em questão pode não ter suporte. Se o objetivo maior é que o uso da tecnologia seja o mais transparente possível conforme descrito nas seções 2.1 e 2.3, é importante que se encontre uma forma de se contornar essa limitação.

A2 QUANTIDADE DE MENSAGENS TRAFEGADAS

O cenário de uso de uma *smarthome* tipicamente está atrelado à utilização de uma rede infraestruturada. Isso permite que o *Daemon* seja configurado para não retransmitir as mensagens recebidas, uma vez que todos os dispositivos estão interligados. Para isso, basta habilitar uma configuração booleana nos *Daemons*

instalados nos dispositivos. A confiabilidade pode ser garantida por protocolos em alto nível se requerido em uma aplicação.

A3 GARANTIA DE ENTREGA DAS MENSAGENS ENVIADAS

Conforme destacado na Seção 4.5.4, não há uma garantia de entrega das mensagens enviadas. Portanto, para um ambiente real de uso, cada aplicação implementada precisa avaliar se as taxas de entregas superiores a 96% avaliadas em [MORAES, H. F. et al., 2012] são suficientes ou se será necessário desenvolver a nível de aplicação uma abstração que garanta a entrega de 100% das mensagens. É importante enfatizar que cada aplicação precisa definir suas respectivas políticas de tratamento de erros uma vez que o desenvolvimento dessas aplicações dependem do tratamento das exceções previstas.

A4 SEGURANÇA

Sob a ótica de segurança, é fundamental que a rede construída esteja configurada com protocolos de segurança como o WPA. Além disso, para aplicações com informações sensíveis, como, por exemplo, uma aplicação na área de saúde como o SCIADS [CARVALHO, S. T. et al., 2010], recomenda-se a utilização de protocolos de segurança análogos ao HTTPS, de forma a garantir que o fato da rede ser colaborativa não permita a visualização indevida de conteúdos relevantes.

A5 QUALIDADE DE SERVIÇO (QOS)

Aplicações reais podem necessitar de um tratamento diferenciado na entrega de mensagens. Se um acidente ocorre ou se uma informação extremamente relevante precisa ser logo transmitida, requisitos de qualidade de serviço tornam-se importantes. Conforme já descrito na Seção 4.5.7, o protocolo não permite nativamente o suporte para tal, ficando a critério da aplicação desenvolver mecanismos de contornar essa limitação.