Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado

Mestrado em Engenharia Informática

Soluções descentralizadas para o encontro de parceiros baseadas em critérios geográficos

João Diogo Nunes Cartaxo (27241)

Lisboa
(2010)

Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado

Soluções descentralizadas para o encontro de parceiros baseadas em critérios geográficos

João Diogo Nunes Cartaxo (27241)

Orientador: Prof. Doutor Sérgio Marco Duarte
Co-Orientador: Prof. Doutor Nuno Manuel Ribeiro Preguiça

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para a obtenção do Grau de Mestre em Engenharia Informática.

Lisboa
(2010)

Para os meus pais, irmão, sobrinho e namorada

Agradecimentos

A redacção desta dissertação é o culminar de um longo e exigente percurso. Contudo, tal não seria possível sem o apoio, amizade, tolerância e compreensão de muitos, a quem não posso deixar de agradecer. Em primeiro lugar, desejo agradecer tanto ao Professor Doutor Sérgio Marco Duarte, como ao Professor Doutor Nuno Manuel Ribeiro Preguiça, pela oportunidade concedida e pela paciência, ajuda, conselhos e orientação ao longo de todo o trabalho.

Agradeço igualmente à minha namorada, Rute Graça, por todo o apoio, compreensão e coragem transmitidos, mas principalmente por todo o amor demonstrado, que contribuiu significativamente para que pudesse manter a concentração e motivação necessárias.

Aos meus pais, irmão e sobrinho, bem como à família da minha namorada, agradeço pela paciência e pelo apoio demonstrados, principalmente nas situações de maior aperto.

Desejo igualmente agradecer a todos as restantes pessoas que, directa ou indirectamente, contribuíram para a realização desta dissertação de Mestrado, onde se destacam:

- Todos os meus colegas e amigos, com os quais convivi e trabalhei: Nuno Luís, João Morais, Pedro Sousa, Ricardo Martins, Rui Domingues, Nuno Boavida, Pedro Bernardo, Danilo Manmohanlal,
- Todos os membros da Hexastep, com quem trabalhei paralelamente ao desenvolvimento da dissertação, em especial ao Henrique Soares e Carlos Almeida pela compreensão e tolerância demonstradas.

A todos, o meu muito obrigado !

Sumário

A computação ubíqua representa uma emergente disciplina da computação distribuída, e passa pela utilização de dispositivos móveis enquanto entidades computacionais capazes. Dentro deste paradigma, enquadram-se diversos outros conceitos derivados, entre eles o *Participatory Sensing*, que consiste na exploração das capacidades sensoriais dos telemóveis e na mobilidade dos utilizadores para realizar tarefas sensoriais colectivas de larga escala.

Tipicamente, as aplicações pervasivas tendem a centralizar a complexidade nas componentes fixas, atribuindo aos dispositivos móveis tarefas de recolha de dados e interacção com os utilizadores. Nesta dissertação pretende-se explorar melhor as capacidades computacionais dos dispositivos móveis, aumentando a sua autonomia de operação. No entanto, para que estes consigam desempenhar um papel mais activo é necessário que consigam encontrar-se mutuamente e comunicar de forma eficiente, respeitando as suas limitações de disponibilidade de recursos.

Na presente dissertação, apresentamos um conjunto de soluções para resolver o problema do encontro mútuo entre entidades móveis num contexto descentralizado de forma dinâmica e eficiente. Estas soluções são discutidas no contexto de um serviço de *car pooling*, mas procura-se que sejam suficientemente genéricas para que possam ser usadas para suportar outras aplicações de *participatory sensing*.

Palavras-chave: *Participatory sensing*, comunicação, boleia, computação ubíqua, telemóvel.

Summary

Ubiquitous computing represents an emergent discipline of distributed computing, and relies upon the usage of mobile devices as capable computational devices. Within this paradigm, there are some derived concepts, such as *Participatory Sensing*, which consists on exploiting the sensing capabilities of modern mobile phones and the mobility of users, to perform large scale sensing tasks.

Typically, pervasive applications tend to centralize the complexity onto the fixed components, whereas the mobile devices are assigned with tasks of data collection and user interaction. This MSc dissertation aims to better exploit the computing capabilities of mobile devices, increasing their operating autonomy. However, to play a more active role they need to be able to find each other and communicate efficiently, respecting their own lack of resource limitations.

In this MSc dissertation, we propose a set of dynamic and efficient solutions to solve the problem of mutual encounter between mobile entities in a decentralized context. These solutions are discussed in a car pooling service context, but we intend to make them sufficiently generic so that they can be capable of supporting other participatory sensing applications.

Keywords: *Participatory sensing*, communication, carpooling, Ubiquitous computing, mobilephone.

Conteúdo

Lista de Figuras

Lista de Tabelas

1

Introdução

1.1 Motivação

A computação ubíqua tem-se vindo a tornar numa área da informática em grande expansão, impulsionada pelas constantes evoluções a nível tecnológico, que tornam os dispositivos móveis cada vez mais sofisticados e pequenos. Como bom exemplo destas evoluções temos os telemóveis, que, nos dias que correm, encontram-se munidos com as mais variadas tecnologias como acelerómetro, câmara fotográfica, WiFi, GPS, Bluetooth, etc, bem como uma crescente capacidade de processamento. Estes tornaram-se portanto em dispositivos atractivos para utilização em sistemas distribuídos ubíquos, beneficiando da vantagem de estarem constantemente presentes no nosso quotidiano, face aos dispositivos habitualmente utilizados neste tipo de sistemas.

Desta forma, é possível desenvolver soluções que assentam no pressuposto de que cada indivíduo tem a capacidade de ter uma participação activa e expontânea, bem como a capacidade de executar processamento de dados. Um exemplo de tais sistemas está presente nas aplicações de *Participatory Sensing* [8], que assentam na utilização de dispositivos móveis pessoais, enquanto instrumentos sensoriais. Mediante estes, os utilizadores podem participar activamente em recolhas de dados mediante um contexto específico, sendo estes dados disponibilizados para toda uma comunidade de utilizadores com interesses comuns.

Porém, a utilização de telemóveis enquanto dispositivos de computação levanta algumas questões, resultantes do facto de se tratarem de dispositivos limitados em termos de disponibilidade de recursos, e com um tempo útil de operação limitado. Desta forma, as questões apontadas prendem-se com a necessidade de desenvolver soluções de comunicação que sejam baratas sem comprometer a eficiência, encontrar formas de garantir a persistência de grandes volumes de dados sem esgotar os recursos disponíveis, entre outras.

Nesta dissertação endereçamos um dos problemas relacionados com a comunicação, mais precisamente o problema do encontro mútuo entre utilizadores que, mediante o contexto e as circunstâncias em que se encontram, devem comunicar. Este problema seria de fácil resolução se existisse a possibilidade de recorrer a uma solução centralizada, ou se existissem os recursos necessários para levar a cabo uma solução de procura exaustiva. Porém, é necessário encontrar uma solução que seja capaz de garantir os graus de eficiência e de eficácia necessários, tendo sempre presente o facto de que a disponibilidade de recursos é limitada.

Assim, pretendemos desenvolver soluções para este problema de forma transparente para as aplicações, resolvendo o problema de forma distribuída e assumindo que não existem referencias pré-acordadas entre os utilizadores.

1.2 Descrição do Problema

A ideia de desenvolver um suporte para a comunicação entre entidades móveis em ambientes pervasivos não é original, pois existem já plataformas de encaminhamento direccionadas para estes ambientes. Porém, estas tendem a centralizar a complexidade na infra-estrutura fixa, atribuindo um papel limitado aos dispositivos móveis, e consequentemente, ignorando as suas capacidades de computação.

Assim, será interessante explorar um cenário em que alguma da complexidade é exportada para os dispositivos móveis, transformando-os em mais do que simples dispositivos de recolha dados por interacção com o utilizador. Desta forma, conseguem-se desenvolver entidades móveis mais autónomas, potenciando uma melhor distribuição de carga geral do sistema distribuído. No entanto, para que consigam tirar proveito desta autonomia, será necessário que as entidades móveis consigam encontrar-se mutuamente, com base em critérios geográficos e temporais, de forma eficiente e barata, sem recorrer a uma entidade central que resolva o problema.

Para tal, e tendo presente o facto de que os dispositivos móveis apresentam ainda limitações energéticas, que restringem o tempo útil de funcionamento, cremos que a

resposta está em criar um ambiente simbiótico entre as entidades móveis e a infra-estrutura fixa. Neste, as entidades constituintes da infra-estrutura fixa oferecem serviços de intermediação e suporte computacional às entidades móveis tanto na resolução do problema do encontro mútuo (entre entidades móveis), como no desempenho de tarefas que se mostrem demasiado complexas para serem executadas nos dispositivos móveis.

Nesta dissertação, o problema é contextualizado numa aplicação de *car pooling*, com o objectivo de enquadrar o problema numa situação com contornos reais. A escolha deste contexto recai sobre o facto de existirem já algumas propostas comerciais baseadas em soluções Web, caracterizadas por serem demasiado estáticas e restritivas, envolvendo demasiado o utilizador. Assim, seria interessante a exploração de um serviço em tempo real, mediante o qual um utilizador submetia pedidos de boleias, obtendo uma resposta de viabilidade num curto espaço de tempo. Pretende-se desenvolver este serviço como uma alternativa descentralizada, que seja baseada nas soluções de comunicação propostas.

1.3 Objectivos

A solução a desenvolver baseia-se no modelo *publish/subscribe*, adaptado a um contexto de computação ubíqua sobre a forma de uma plataforma denominada MEEDS. Neste contexto, a solução deverá permitir enviar mensagens para nós com o mesmo conjunto de interesses ou localizados num dado espaço geográfico.

Sumariamente, podemos enunciar o conjunto de objectivos a que nos propomos nesta dissertação:

- Desenvolver um conjunto de soluções sobre MEEDS direccionadas para o problema do encontro entre entidades móveis em ambientes ubíquos distribuídos;
- Implementar um serviço de *car pooling* que opere sobre as soluções desenvolvidas, por forma a contextualizar o problema num caso prático;
- Adaptar o simulador de MEEDS para suportar o serviço de *car pooling* a desenvolver;

1.4 Principais Contribuições

Nesta secção são apresentadas as principais contribuições deste trabalho. Estas são:

- Desenvolvimento de um conjunto de soluções direccionadas para o problema do ponto de encontro entre parceiros num contexto descentralizado. Estas soluções consideram os seguintes cenários:
  - Num primeiro cenário, considera-se o problema num contexto em que todos os parceiros são estáticos, e em que os seus interesses se reflectem sobre zonas geográficas bem definidas e inalteráveis.
  - No segundo cenário considera-se a mobilidade de parte dos parceiros, reflectindo-se esta mobilidade na definição dos critérios geográficos que reflectem zonas de interesse. Assim, o problema é desenvolvido sobre um contexto em que parte dos parceiros é móvel e outra é estática.
  - No terceiro e último cenário considera-se a mobilidade de todos os parceiros, pelo que todas as zonas geográficas definidas reflectem interesses de uns parceiros sobre a mobilidade de outros.
- Como forma de aplicar as soluções propostas, foi desenvolvido um protótipo funcional que implementa um serviço de *car pooling* desenvolvido sobre os 3 cenários descritos. O desenvolvimento deste protótipo envolveu o incremento de ferramentas ao simulador de MEEDS, respectivamente a introdução do conceito de estrada e a possibilidade de configurar a velocidade de deslocação das entidades móveis.
- Sobre o protótipo descrito, foi desenvolvido um módulo de validação que serviu de base à avaliação das prestações das soluções desenvolvidas.

1.5 Estrutura do Documento

O documento é constituído por 6 capítulos:

- **Capítulo 1:** Referente à introdução, onde são apresentadas o motivação e a descrição do problema abordado ao longo da dissertação;
- **Capítulo 2:** São analisados alguns dos projectos que constituem o trabalho relacionado com a matéria da dissertação;

- **Capítulo 3:** Será feita a descrição e análise do problema abordado na dissertação, decompondo-o em diversos cenários sobre os quais são desenvolvidas as soluções propostas;
- **Capítulo 4:** Será descrita em detalhe a plataforma de comunicação sobre a qual são implementadas as soluções descritas no capítulo anterior, bem como são apresentadas as arquitecturas e modelos funcionais de cada uma destas;
- **Capítulo 5:** Descrevem-se as métricas utilizadas para validação das soluções propostas, bem como é feita a análise de resultados obtidos;
- **Capítulo 6:** São apresentadas as conclusões alcançadas com o desenvolvimento da dissertação;

2

Trabalho relacionado

O problema de providenciar uma infra-estrutura de comunicação em ambientes ubíquos tem vindo a ser endereçado em diversos projectos, ainda que nem sempre de forma directa. No presente capítulo são descritos e analisados alguns dos projectos mais relevantes que abordam a temática em causa.

O capítulo encontra-se organizado em 3 principais secções:

- **Aplicações relacionadas:** São analisados os trabalhos que propõem aplicações completas, apresentando objectivos a servir pela aplicação e a respectiva solução de suporte;
- **Arquitecturas e plataformas de comunicação:** São apresentadas plataformas idealizadas para o suporte de aplicações ubíquas genéricas, bem como trabalhos que propõem tanto arquitecturas de rede como protocolos para resolver os problemas levantados pela comunicação em ambientes pervasivos;
- **Balanço crítico:** É efectuado um levantamento de críticas encontradas aos projectos analisados, tendo por base o objectivo subjacente à presente tese;

2.1 Aplicações Relacionadas

As aplicações ubíquas situam-se frequentemente entre dois extremos: As que têm suporte de uma entidade estática, funcionando em modo cliente/servidor; e as descentralizadas, sendo os próprios nós responsáveis pela manutenção da rede distribuída - *peer to peer*.

2.1.1 Cliente Servidor

2.1.1.1 CARTEL

O *CarTel* [24] é um sistema de computação móvel e ubíqua, enquadrado num contexto de colheita e processamento de dados numa rede formada por nós móveis a executar em veículos. Proposto pelo MIT, o sistema pretende disponibilizar uma plataforma capaz de suportar pedidos de utilizadores relativos a informação contextual de uma determinada zona geográfica. Estes pedidos são encaminhados para nós móveis localizados em veículos, que se responsabilizam pela respectiva colheita de dados.

O protocolo de comunicação, é um dos pontos de interesse do *Cartel*, pois procura métodos eficientes para que as comunicações sejam fiáveis.

É utilizada uma arquitectura centralizada, figura 2.1, sendo o ponto central composto por um servidor - **portal**, responsável não apenas pela coordenação e controlo do sistema distribuído, mas também pela disponibilização de um ponto de acesso WEB. Os dispositivos instalados nos veículos, funcionam como nós da plataforma, interagindo enquanto clientes com o portal.

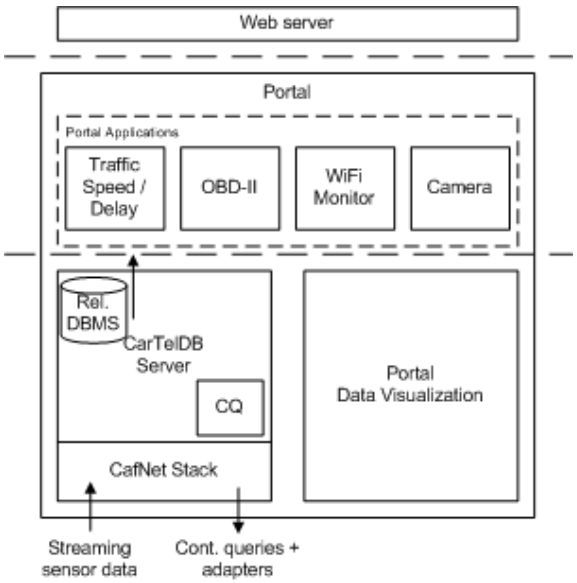


Figura 2.1: Arquitectura Cartel

O *Cartel* suporta igualmente um funcionamento segundo modelos P2P para oferecer suporte a períodos de desconectividade. Esta capacidade é garantida uma vez que o sistema opera, sobre o *Cafnet* [9], apresentado na secção 2.2.2. O *Cafnet* é uma pilha de rede tolerante a atrasos, capaz de implementar políticas de *buferização* e prioritização de pedidos, segundo as instruções da aplicação.

Após os clientes se conectarem ao portal, estes interagem com as aplicações através de GUIs disponibilizadas para o efeito, originando a formulação de pedidos. Estes são processados como *queries* contínuas aos nós móveis, recorrendo a uma API exportada pelo ICEDB [36]. Nestas *queries* encontram-se especificados diversos parâmetros, como os tipos de sensores a utilizar ou tipos de amostragem, consoante o teor dos dados a recolher. Após a recepção do pedido e consequente recolha dos dados pelos dos dispositivos móveis, estes são enviados para o portal. Após recebidos, os dados são armazenados na base de dados suportada pelo ICEDB, ao invés de serem imediatamente devolvidos à aplicação que os requisitou. Desta forma, para recolher os dados amostrados, as aplicações efectuam *queries* locais à base de dados onde estes se encontram. É importante frisar que não existe sincronismo entre o envio de dados por parte dos nós móveis e as *queries* locais por parte da aplicação, pelo que estas *queries* funcionam sobre os dados disponibilizados no instante em que estas são efectuadas.

Apesar de não ser idealizado para telemóveis, o certo é que essa seria uma realidade perfeitamente concebível, uma vez que a capacidade de processamento dos dispositivos móveis utilizados na implementação do sistema estão ao alcance, ou são até mesmo superados pelos telemóveis vulgarmente comercializados.

2.1.1.2 MICRO-BLOG

O Microblog [20] assenta na motivação de criar um mapa virtual do mundo real a partir das capacidades sensoriais dos telemóveis. Este consiste na recolha de dados sensoriais relativos a um ambiente contextual específico (Ex: recolha de imagens de uma rua, medição da força de sinal Wi-Fi num espaço público, etc.), constituindo um microblog. Os microblogs, uma vez criados, são disponibilizados através do Google Maps [21], um sistema de consulta de mapas reais a partir do computador. Estas recolhas são solicitadas por utilizadores que pretendam obter de antemão informação relativa a imagens, cobertura Wi-Fi, ou qualquer outra informação que considerem pertinente relativamente a um local que pretendam visitar. Após a recolha dos microblogs, estes são

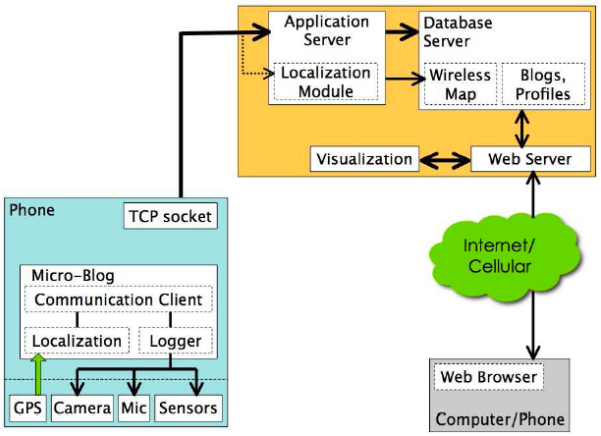


Figura 2.2: Arquitectura Micro-blog

armazenados numa base de dados alojada num servidor, para que sobre eles possam ser efectuados pedidos.

O sistema assenta sobre uma arquitectura cliente-servidor centralizado, figura 2.2, sendo o servidor é responsável pela gestão dos telemóveis, tanto a nível de comunicações como a nível de localização (*Application Server*); pela gestão do armazenamento dos microblogs e dados relativos tanto a telemóveis como a informações sensíveis dos utilizadores do sistema (*Database Server*); e pela disponibilização aos utilizadores da plataforma de interacção WEB (*WEB Server*).

Para efeitos de localização dos telemóveis, o sistema mantém uma rotina periódica, tendo primordialmente em conta as necessidades energéticas que este processo acarreta. Se este fosse efectuado apenas com recurso ao GPS, apesar de obter leituras com uma margem de erro muito diminuta, isso implicaria um consumo energético capaz de incapacitar a bateria do telemóvel em apenas 7h. Assim, para conseguir um melhor aproveitamento energético, o sistema procura a alternância entre tecnologias e protocolos de localização [11].

Uma vez recebidos os pedidos dos utilizadores através da interface WEB, o sistema efectua uma avaliação sobre as intenções do utilizador, ao confirmar a existência prévia de microblogs passíveis de responder às suas necessidades. Não sendo possível a formulação de uma resposta, o pedido é encaminhado para telemóveis localizados na zona sobre a qual se pretende a recolha de dados. Dado que os telemóveis notificam periodicamente o servidor da sua localização, isto torna possível que qualquer telemóvel que entretanto alcance a área sobre a qual é efectuado o pedido de recolha de dados participe na recolha. Quando um telemóvel responde ao pedido, a resposta é associada ao pedido, armazenada na base de dados (como microblog) e colocada/actualizada no mapa.

Para a privacidade dos utilizadores dos telemóveis participantes em recolhas, os autores do sistema propõem a criação de diferentes níveis de operação. Estes diferem entre si relativamente à granularidade de informação sensível ao utilizador que é visível. No entanto, apesar de ser uma critica especulativa, não é complicado inferir que muitos utilizadores se mostrem renitentes à operação em modo completamente "aberto", pois isso implica uma certa dose de altruísmo por parte dos mesmos, uma vez que as mais valias oferecidas pelo sistema, podem ser em grande parte satisfeitas pelos restantes níveis de privacidade.

2.1.1.3 POTHOLE PATROL

Desenvolvido pelo MIT *Computer Science and Artificial Intelligence Laboratory*, o *Pothole Patrol (p^2)* [18] pretende oferecer um sistema de monitorização e mapeamento das condições de conservação de estradas. O projecto tem como objectivo, não apenas o benefício público de alerta aos automobilistas relativamente às melhores opções rodoviárias a tomar, mas também num sentido de construir um sistema que possibilita a criação de um suporte de prioritização de estradas a reparar.

A arquitectura do sistema assenta sobre um modelo cliente-servidor, segundo o qual os veículos equipados com acelerómetros, GPS e dispositivos WiFi executam o papel de clientes, interagindo com um servidor estático. As comunicações são processadas através de WiFi, fazendo recurso dos protocolos de *opportunistic wireless communication* também observados no *CarTel* [24], ou utilizando a rede telefónica. Para lidar com a possibilidade de ocorrência de períodos de desconexão, o *Pothole Patrol* implementa um sistema de *bufferização* e transmissão fiável de dados: *dPipe*, sendo este sistema uma extensão conceptual da abstracção *Pipe*, existente nos sistemas UNIX, que permite a comunicação fiável e tolerante a atrasos de entre processos localizados em hosts distintos.

Neste projecto, o foco principal centra-se não no processo de comunicação mas no processo optimização de colheita e filtragem de dados. As leituras dos pavimentos são efectuadas através de acelerómetros capazes de fornecer dados nos 3 eixos cartesianos, complementados com a localização e estampilha temporal obtidas a partir do GPS. Porém, o sistema necessita de uma forma eficaz de interpretação dos dados obtidos sendo necessária uma contextualização das leituras. Esta necessidade surge pela existência de situações alheias ao problema que podem interferir com as leituras (Ex: o fechar das portas, influências derivadas da condução, etc.). O facto de as leituras poderem sofrer ligeiras alterações em amostragens distintas acentuam mais esta necessidade, pois esta situação pode dar azo a que uma única anomalia surja no sistema como um conjunto distinto de anomalias. Para dar resposta a esta necessidade, o *Pothole Patrol* sujeita os sensores a um período de aprendizagem, funcionando através de inserções manuais de ocorrências ao circular com o veículo por zonas passíveis de conflito (Ex: Passagem por um buraco - anomalia, ou por uma linha de ferroviária - benigno).

Para evitar leituras correspondentes a falsos positivos, o sistema aplica um sistema de filtragem complexo sobre as leituras efectuadas. Uma vez completo este processo, os dados resultantes são sintetizados e apresentados num mapa, sendo posteriormente disponibilizado via WEB com recurso a plataformas pré-existentes para o efeito [21].

2.1.1.4 BUBBLE-SENSING

O Bubble-Sensing [23] assenta sobre o conceito de recolha sensorial por meio de telemóveis para disponibilizar um sistema de mapeamento de tarefas por zonas geográficas concretas. Um utilizador poderá, por exemplo, estar interessado em obter uma recolha de amostras sonoras efectuadas sobre uma localização de seu interesse. Assim, o sistema permite a criação de uma "bolha geográfica"(*bubble*), mantida pelos dispositivos móveis que se encontrem na zona.

O sistema baseia-se na ideia chave de que apesar de ser impossível controlar a mobilidade dos indivíduos, esta não se desenvolve de forma aleatória, pois um ponto de interesse (num espaço público) para um indivíduo também o será, com grande probabilidade, para diversos outros indivíduos.

O sistema é construído sobre um modelo cliente/servidor centralizado, utilizando comunicação através de WiFi. O servidor é responsável por manter a base de dados relativamente às zonas sobre as quais existem tarefas. A criação de tarefas pode ser efectuada através da interface Web, ou através do telemóvel, deslocando-se o indivíduo para o local pretendido e iniciando o processo de criação de tarefa. Os dados recolhidos são posteriormente disponibilizados ao utilizador que deu origem à criação da tarefa. Os clientes são representados por aplicações a correr nos telemóveis, e podem tomar diferentes papéis consoante as capacidades e configurações do mesmo. Todos os telemóveis localizados na zona são potenciais nós de recolha - ***sensing nodes*** a menos que não possuam o hardware necessário para tal. Quando um utilizador cria uma tarefa sobre uma zona através do telemóvel, fornecendo a localização e o tempo durante o qual a tarefa deve ficar activa, esta fica-lhe associada como sendo o ***bubble creator***. Neste lugar, o utilizador fica encarregue de notificar os utilizadores na sua vizinhança da criação da tarefa, iniciando um processo de *broadcast* para publicar a especificações da tarefa. No caso de existir conectividade, o *bubble creator* notifica também o servidor.

Dada a mobilidade dos utilizadores, as “bolhas” podem sofrer desvios relativamente à zona de interesse. Para colmatar este comportamento, um utilizador pode tomar o papel de ***bubble anchor*** ficando responsável por efectuar *broadcast* de *beacon messages* que desempenham o mesmo papel das mensagens enviadas pelo *bubble creator*. A eleição do *bubble anchor* é efectuada tendo em conta ou a mobilidade ou a localização dos utilizadores, sendo este um processo probabilistico. Assim, são eleitos os utilizadores que ou se encontram mais perto do centro da zona de interesse, ou que apresentam menor taxa de mobilidade ao longo do tempo, sendo o processo de eleição repetido até atingir um máximo de envio de *beacon messages*.

Uma vez que existe a possibilidade de ausência de utilizadores na zona de interesse, é necessária uma forma de restauro da tarefa sendo esta uma responsabilidade dos ***bubble carriers***. Estes são providos de capacidade de auto-localização e conectividade ao servidor, efectuando *updates* periódicos da sua localização e requisições de tarefas existentes para a sua presente localização. Se um *bubble carrier* visita uma zona para a qual existem tarefas, e não recebe *beacon messages*, então este simula o papel de *bubble creator* até que um novo *bubble anchor* seja eleito ou até que termine o período de actividade da tarefa.

Seria interessante adicionar ao projecto um mecanismo de detecção de ausências. Este mecanismo teria por função detectar a possível movimentação do utilizador para fora da "bolha", tomando a iniciativa de anunciar a sua saída aos demais nós na zona. Desta forma, cada nó poderia manter um número estimativo de utilizadores na zona, e sempre que oportuno, anunciar este número ao servidor, no sentido de movimentar "reforços" para as zonas mais desprovidas de *sensing nodes*.

2.1.2 *Peer To Peer*

2.1.2.1 MOBEYES

O MobEyes [28] pretende ser um sistema de vigilância activa de actividades criminosas/suspeitas através do uso de veículos. Para tal, o sistema recorre a uma rede *ad hoc* de sensores móveis alojados nos veículos participantes. Desta forma, transformam-se os veículos em agentes monitores, conhecidos no artigo como *Smart Mobs*, que oferecem suporte à construção de uma base de dados de documentação preventiva das referidas actividades ilícitas.

Os dados recolhidos são processados/filtrados localmente nos veículos, com vista a reduzir substancialmente a dimensão da informação a transmitir. Dada a reduzida dimensão dos sumários resultantes da fase de processamento, o sistema é capaz de encapsular numa única mensagem vários destes sumários, melhorando consequentemente o desempenho do sistema. Isto é importante, pois as conexões entre veículos são de curta duração, devido ao movimento.

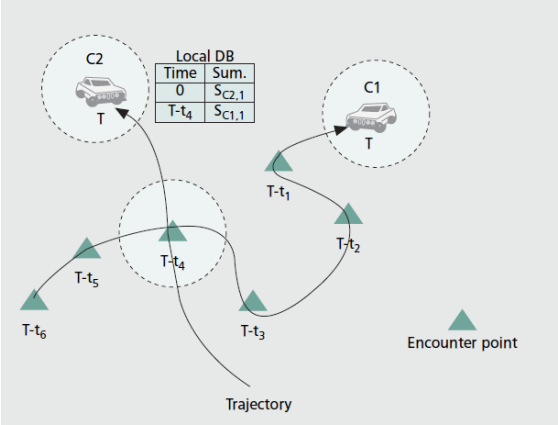


Figura 2.3: Mobeyes *Data Mulling*

No cenário MobEyes não existe uma verdadeira urgência na transmissão dos dados, pelo que as mensagens são transmitidas utilizando o conceito de Data Muling [35]. Concretamente, os veículos, após efectuarem a recolha de dados, armazenam-nos localmente até que possam ser transmitidos a outros veículos por difusão passiva, ao entrar no raio de alcance dos mesmos. Uma vez presentes as condições necessárias para o estabelecimento de uma comunicação entre veículos, o sistema efectua a transmissão de sumários baseando-se em heurísticas bem definidas. Segundo estas, as mensagens são transmitidas apenas uma vez entre os veículos participantes na conexão (single-hop) ou podem ser re-encaminhadas para outros veículos em comunicações posteriores (multi-hop). Para efeitos de colheita de sumários, as entidades responsáveis (idealmente veículos de agentes policiais) efectuam um processo de inquisição sobre veículos com os quais contactam ao longo dos seus percursos, enviando mensagens de *Harvest*. Nestas mensagens, as entidades responsáveis enviam um *Bloom Filter* [15] que o veículo utiliza como filtro probabilistico a aplicar aos sumários a enviar, por forma a eliminar o envio de informação redundante (repetida). Uma questão pertinente que é omissa no artigo, refere-se ao "garbage collection"destes sumários.

2.1.2.2 MOBILE CHEDAR

O Mobile Chedar [27], apresenta uma extensão ao Chedar (cheap distributed architecture) [2], uma camada de middleware desenvolvida para aplicações P2P a operar sobre a WEB. Esta camada permite o desenvolvimento de aplicações de partilha de recursos (ficheiros, CPU, etc.) segundo uma API bem definida. O objectivo da extensão proposta é acrescentar conectividade com nós móveis, explorando a ubiquidade dos mesmos, figura 2.4.

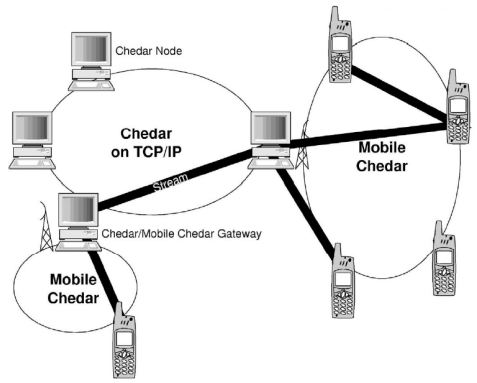


Figura 2.4: Mobile Chedar *overview*

A comunicação móvel entre os nós participantes é efectuada através de Bluetooth, o que implica a proximidade dos todos dispositivos. Assim, para fazer a transição entre o ambiente estático da WEB e a mobilidade dos dispositivos móveis, os *gateways* de Chedar incorporam adaptadores bluetooth. A partir destes, os *gateways* participam nas piconets enquanto *peers*, e funcionam como ponto de conexão da rede móvel para a rede chedar.

Para disponibilizar o serviço de partilha de recursos, o sistema adopta um modelo de *System Multicasting* [12]. Segundo este, os nós da rede chedar disponibilizam os recursos através de *streams*. Quando um nó móvel pretende usufruir desses recursos, efectua um *subscribe* da *stream*. O sistema possibilita que vários nós subscrevam uma *stream*, sendo que após a subscrição, o nó passa igualmente a fazer *publish* da mesma. Assim, para subscrever uma *stream* que disponibilize um serviço pretendido, basta localizar um dos nós que a disponibilizem.

A descoberta de nós é feita sobre uma política de *1-hop* queries, segundo a qual, os nós móveis enviam um pedido para todos os seus nós vizinhos ao alcance. Se o pedido for recebido pelo nó *gateway*, este encaminha-o, juntamente com um TTL para todos os seus nós vizinhos da rede chedar. Se um dos nós da rede chedar for capaz de dar resposta a um dos recursos solicitados, este responde ao nó de onde recebeu o pedido, e a resposta segue o caminho inverso ao do pedido até ao nó móvel. Uma vez reunidas as localizações dos recursos, os nós móveis disponibilizam esta informação à aplicação, que se encarrega da decisão final.

Dada a limitação de proximidade do Bluetooth, é fácil antever a fraca aplicabilidade em ambientes urbanos públicos. Porém, para contrapor esta tendência poder-se-ia optar pela recorrência ao WiFi, assumindo no entanto a existência de redes públicas abertas.

2.2 Arquitecturas e Plataformas de comunicação

2.2.1 DEEDS

Deeds [16], consiste numa plataforma de disseminação de eventos extensível e distribuída, que assenta num conjunto de modelos de disseminação orientados aos canais de eventos com garantias de qualidade de serviço (QoS) genéricas (sobretudo qualitativas), tais como fiabilidade, ordem, persistência, etc. Estes modelos procuram simplificar o desenvolvimento das aplicações, ao permitir que estas seleccionem os tipos de canais de eventos com a QoS mais apropriada às suas necessidades. Dessa maneira, limita-se o código que é necessário colocar na aplicação para aproximar a QoS oferecida pelo substrato de disseminação daquela que é realmente exigida. Por outro lado, existe ainda um potencial ganho de eficiência, pois esta abordagem poderá evitar que as aplicações paguem desnecessariamente o custo adicional associado ao suporte de QoS excessivas. Isto é algo que tenderá a ocorrer se a QoS de referência do substrato

de disseminação for nivelada por cima.

Para suportar os referidos objectivos, o sistema opera sobre o conceito de canais de eventos activos. Estes são estruturados mediante uma rede lógica activa, sendo esta extensível relativamente às capacidades de comunicação. É sobre esta capacidade de extensibilidade que assenta o teor da presente dissertação, sendo objectivo desta a exploração de combinações interessantes de canais de eventos e mecanismos de comunicação - transportes. As referidas combinações têm por base o não só o suporte à aplicação de *car pooling*, como também a sua simplificação arquitectural, factorizando as complexidades inerentes com garantias de qualidade de serviço para a plataforma, com vista à sua reutilização por outras aplicações do mesmo paradigma.

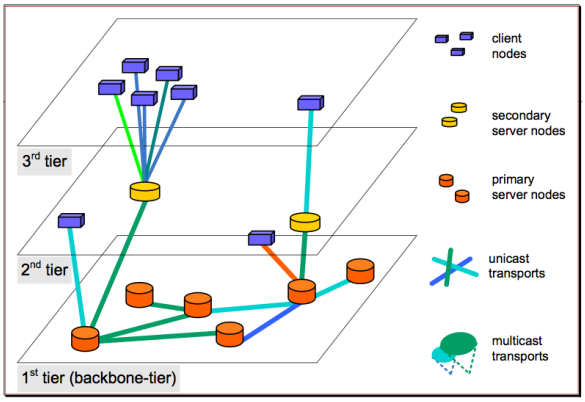


Figura 2.5: DEEDS: Modelo *3 tier* de redes sobrepostas

2.2.2 CAFNET

O *CafNet* [9] consiste numa camada de rede tolerante a atrasos e longos períodos de desconectividade. Este serviço responsabiliza-se por atender os pedidos de comunicação por parte da camada aplicacional, e é oferecido na forma de uma rede sobreposta (*Overlay Network*), construída sobre o nível de suporte aos protocolos de comunicação. Isto torna possível controlar o modo de envio das mensagens submetidas pelas aplicações, bem como controlar o uso dos mecanismos de comunicação.

O *CafNet* encontra-se subdividido em 3 níveis sobrepostos, figura 2.6: ***CafNet Transport Layer* (CTL)**, responsável por comunicar com o nível aplicacional, recebendo os pedidos directamente e reencaminhando as respostas para o mesmo.

Esta camada implementa ainda um sistema de reenvio periódico de mensagens para todas as aplicações que se mantenham na lista de *unacknowledged* para além de um período de tempo. Todos os pedidos recebidos por esta camada são transmitidos para a camada seguinte: ***CafNet Network Layer* (CNL)**, responsável quer pela *bufferização* de mensagens, quer pela gestão de nomes relativos aos *endPoints* vizinhos com os quais pode comunicar. Assim que seja possível estabelecer uma comunicação com o *endPoint* de destino, as mensagens são passadas ao último nível: ***Mule Adaptation Layer* (MAL)**, que se encarrega do envio da mensagem através do estabelecimento de

uma comunicação com a MAL do *endPoint* de destino, recorrendo a um protocolo semelhante ao HTTP para efectuar a troca de mensagens. Este nível é ainda responsável por determinar os vizinhos do dispositivo em cada instante, notificando o CNL sempre que se denotem alterações. Ao encontrar um novo vizinho, o CNL envia uma mensagem de auto-identificação, sendo portanto o sistema responsável por se dar a conhecer (anunciar) aos demais vizinhos. É importante frisar que cada protocolo de comunicação suportado no nível de rede sobre o qual assenta da pilha de execução do *CafNet* implica uma camada MAL, isto é, existirão tantas camadas MAL quantos os protocolos suportados pelo nível rede que suporta o sistema.

A comunicação efectuada para as camadas superiores é desenvolvida tendo por meio o uso de *upcalls* ou *callbacks* [13], uma vez que não existem garantias de entrega imediata de mensagens. Assim sendo, o MAL efectua *callbacks* sobre a camada CNL para realizar as notificações de novos vizinhos. O CNL efectua *upcalls* sobre o CTL para notificar da existência de espaço de *bufferização* livre. Por sua vez, o CTL efectua *upcalls* sobre as aplicações a correr sobre o *CafNet* para fazer o pedido de dados a transmitir.

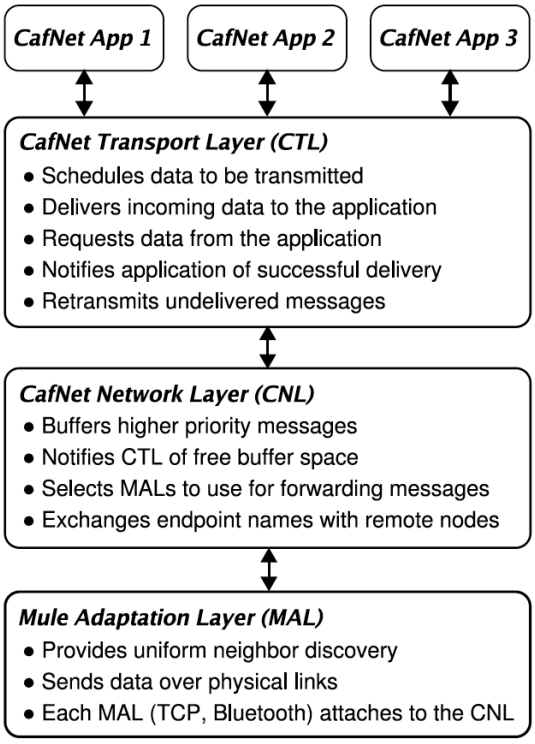


Figura 2.6: Cafnet: Níveis funcionais

Uma vez que um dos requisitos do *CafNet* é o suporte a períodos indeterminados de desconectividade, este faz uso de conexões baseadas em mensagens (*message oriented*) ou invés de conexões baseadas em canais de comunicação (*stream oriented*). A razão para tal baseia-se no facto de que esta arquitectura de comunicação apresenta um melhor suporte para conexões de elevada latência [19]. Para efeitos prioritização, todas as mensagens geradas no CTL são providas de um valor de prioridade, sendo este factor necessário dada a elevada probabilidade de ocorrerem atrasos no envio de mensagens.

2.2.3 CABERNET

O Cabernet [17] é um sistema de comunicação entre veículos em mobilidade, e opera sobre WiFi, explorando redes abertas encontradas oportunisticamente. O sistema pretende oferecer uma plataforma de comunicação transparente para a rede, centrando a complexidade de implementação de protocolos no lado do cliente, que opera sobre veículos.

O objectivo do sistema consiste no desenvolvimento de técnicas de diminuição do tempo de estabelecimento de ligações, e no suporte a períodos de desconectividade.

Para alcançar os objectivos apresentados, o Cabernet comporta duas fases distintas. A primeira fase consiste no estabelecimento e gestão de conectividade ficando esta fase a cargo do *QuickWiFi*. A segunda fase consiste na transmissão de dados mediante um protocolo de comunicação próprio - CTP.

O *QuickWiFi* mantém uma pesquisa continua por *Access Points (AP)*, tentando estabelecer uma ligação assim que encontre um AP que reuna as condições necessárias para tal (Rede aberta e conectividade *end-to-end*). Reunidas as condições, o *QuickWiFi* executa o mesmo processo de estabelecimento de ligação observado na redes WiFi, introduzindo no entanto algumas modificações. Deste conjunto de modificações fazem parte a redução dos *timeouts* das mensagens trocadas, o ignorar de mensagens de *ack* da rotina de autenticação (uma vez que nas redes abertas, esta passo sucede sempre), e a implementação de rotinas capazes de reagir rapidamente quando a conexão é perdida. Porém, a diminuição dos *timeouts* das mensagens poderá acarretar problemas de congestionamento dos APs com o crescimento da escala do sistema. Este problema poderá advir do aumento no número de veículos num dado instante a entrar no alcance de um AP.

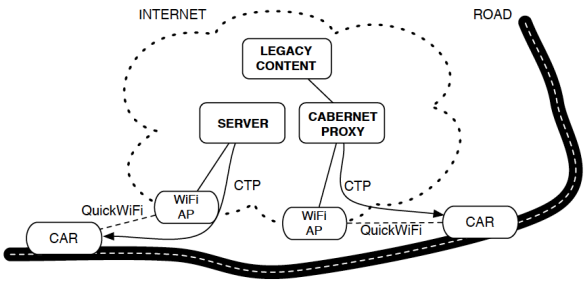


Figura 2.7: Arquitectura Cabernet

Para lidar com a intermitência de conectividade, o sistema contempla a inclusão de uma *proxy* entre os veículos e qualquer *host* Web, figura 2.7. Este é responsável por fazer a transição de mensagens entre o protocolo de comunicação CTP e os demais protocolos WEB (HTTP, TCP, etc.), quer por garantir transparência dos períodos de desconectividade para os veículos, garantindo a persistência do IP entre os mesmos períodos.

A razão de ser da criação tanto do CTP como da *proxy* reside na impossibilidade de manipular os APs, pois todas as componentes de investigação analisadas no artigo sobre introdução de melhoria através do protocolo TCP [3–5,7,10] implicam essa capacidade. O CTP pretende ser fiável e orientado à conexão, destacando-se do TCP pela persistência de sessões entre períodos de desconectividade. Esta propriedade é conseguida através da atribuição de identificadores independentes da rede aos veículos. O CTP consegue igualmente efectuar distinção entre congestionamento da rede e percas de pacotes através de WiFi, reagindo unicamente ao congestionamento. Esta capacidade é conseguida através do envio periódico, por parte dos veículos, de mensagens de *probing*, sondando o estado da rede. A ausência de resposta será interpretada como sinal de congestionamento.

A escolha dos *Access Point* com o qual os veículos se ligam poderia ser um alvo de investigação posterior, pois uma escolha simplista não será possivelmente sempre a melhor opção. Poderá compensar a perca de tempo na análise das forças dos sinais das redes alcançáveis, e tentar o estabelecimento com a rede que apresentar o melhor sinal, garantindo desta forma uma maior fiabilidade de conexão.

2.2.4 Exploiting Our Computational Surroundings for Better Mobile Collaboration

O objectivo dos autores do artigo [6], consiste no desenvolvimento de um protocolo de replicação capaz de explorar a ubiquidade de nós alheios ao sistema. O intuito consiste em aumentar quer a performance quer a disponibilidade do protocolo de anti-entropia, respeitando os recursos de nós explorados. Distinguem-se duas classes de participantes: **nós réplica**, que constituem nós do sistema de replicação; **nós não réplica**, desprovidos de réplicas, são explorados para efectuar comunicação indirecta entre os nós réplica.

O objectivo é conseguido através da optimização da fase de estabelecimento de acordo de *updates* entre nós réplica. Esta optimização consiste na omissão de informação passível de descartar, transmitindo apenas os metadados necessários para o funcionamento das restantes fases do protocolo. Os metadados são transportados pelos nós não réplica em mensagens de consistência. Desta forma permite-se que dois nós réplica comuniquem indirectamente para o estabelecimento de uma rotina de *updates*.

A recorrência a mensagens de consistência numa fase *pre-commitment*, permite aceleração da fase de acordo, pois quanto mais rápida for esta fase, mais rapidamente se atingirá a fase de *commitment*, e menor será a dimensão da janela temporal em que

podem ocorrer *hidden conflicts*. As mensagens de consistência permitem igualmente prevenir contra conflitos de concorrência, uma vez que permitem o adiar de escritas até à chegada dos *updates* concorrentes, evitando desta forma *aborts* desnecessários.

Ainda que nada ser referido relativamente à problemática de comunicação, esta constitui um pressuposto que deverá ser garantido com o mínimo de falhas possível. Porém, dada a reduzida dimensão das mensagens de consistência, a adaptabilidade a protocolos de comunicação por mensagens fica garantida.

O protocolo apresentado constitui uma alternativa viável a utilizar em ambientes computacionais temporariamente isolados, isto é, onde qualquer conexão Web é substancialmente susceptível a falhas de duração indeterminada. Isto porque permite que os nós participantes num qualquer sistema que opere sobre esta arquitectura, consigam estabelecer um consenso entre si sem necessidade de intervenção de um ponto centralizado (servidor). Desta forma, podem-se criar redes temporárias entre nós participantes num qualquer sistema aplicacional comum, que permitem o continuar de tarefas pendentes.

2.2.5 HAGGLE

O Haggle [34] propõe uma plataforma de comunicação capaz de operar eficientemente num ambiente de conectividade intermitente. A necessidade para tal plataforma é criada pela "rigidez"imposta pelos modelos de rede tradicionais aplicados à WEB, desenhados à volta de uma infra-estrutura física de suporte. Existem diversos factores num ambiente pervasivo, que podem ser explorados em prol da operacionalidade dos nós móveis, que não são suportados por estes modelos. A impossibilidade de suporte a acções assíncronas, onde um nó poderia submeter uma acção sem ter que manter a ligação activa até obter os resultados constitui um exemplo. A interacção directa com nós vizinhos constitui igualmente um factor de interesse para a mobilidade, que dificilmente é suportado pelos modelos tradicionais.

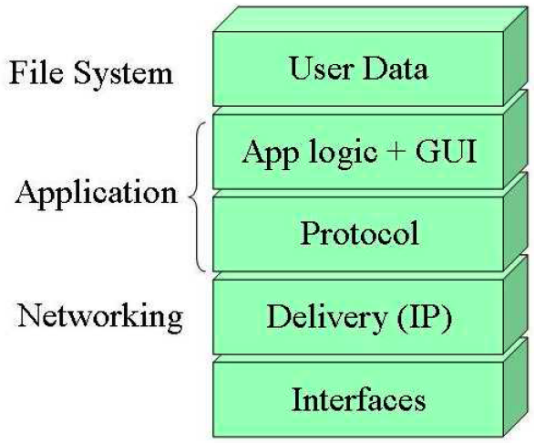


Figura 2.8: Arquitectura de rede corrente para aplicações móveis

O objectivo é conseguido através da exploração de uma política de comunicação oportunistica. O sistema propõe uma arquitectura que centra a complexidade de comunicação num alto nível. Para isto, "encapsula"as camadas aplicacionais e de rede dos protocolos tradicionais, figura 2.8, numa única camada intermediária que faz a transição entre o nível lógico (GUI) e as interfaces de rede.

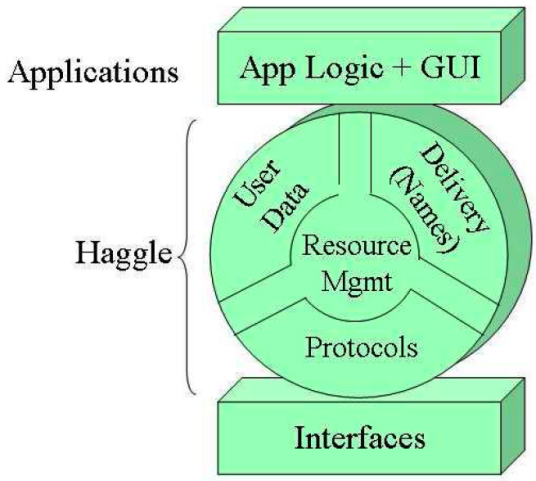


Figura 2.9: Arquitectura proposta: Haggle

Esta camada é composta por quatro módulos lógicos, figura 2.9: *Delivery*: efectua a transformação de nomes em endereços de encaminhamento de mensagens; *user data*: responsável pela gestão dos dados de utilizador; *protocols*: responsável pelo encaminhamento de mensagens; *resource management*: efectua uma gestão energética do dispositivo mediante informação contextual;. Ao albergar todos estes módulos dentro de um único *"framework"*, o sistema consegue fornecer um contexto computacional resiliente, uma vez que controla todo o *middleware* operacional. Assim, não obriga as aplicações a implementarem APIs de comunicação específicas mediante os protocolos usados, conseguindo uma independência entre a aplicação e os protocolos de rede.

O Haggle funciona sobre ADUs (*Application Data Units*) que constituem unidades de encapsulamento de dados. Estas unidades contêm, além conteúdo da mensagem, atributos descritivos constituídos por um par tipo-valor. O tipo descreve o atributo e o valor contém a informação referente para o ADU em causa. Assim, cada bloco de dados é descrito por um conjunto de metadados, tornando a criação de cabeçalhos de ADUs dinâmica. Recorrendo aos ADUs, os nós conseguem fazer *caching* de dados recebidos (por exemplo conteúdo de uma página WEB), ficando aptos a dar respostas a pedidos sobre esses dados.

Apesar de apresentar os conceitos sobre os quais funciona, não existe informação detalhada relativamente à forma como é feita a coordenação dos módulos descritos, tornando-se assim complicado avaliar a viabilidade da arquitectura.

Aplicado à problemática abordada na presente dissertação, a estipulação de unidades de comunicação como os ADUs seria uma mais valia possível, uma vez que possibilita o estabelecimento de atributos. Esta estipulação poderia ser explorada no

sentido de criar *templates* de pedidos de boleias, estipulando todos os dados necessários nos atributos. Desta forma, seria possível fazer uma reutilização de ADUs, bastando apenas efectuar alterações a nível dos atributos, podendo acrescentar ou remover informação descritiva.

2.2.6 SENSEWEB

O SenseWeb [22, 29], consiste numa infra-estrutura de suporte à partilha de sensores, possibilitando a partilha generalizada dos mesmos, a qualquer tipo aplicações que deles necessitem. Este assenta num conceito de "sociedade virtual", onde cada participante contribui para o sistema num intuito de proveito generalizado. Assim, cada participante é um potencial fornecedor de sensores, podendo disponibilizar os seus dispositivos sensoriais na rede. As aplicações ligadas à infra-estrutura, podem recorrer a estes sensores para obterem recolhas de dados sobre os quais necessitem de operar.

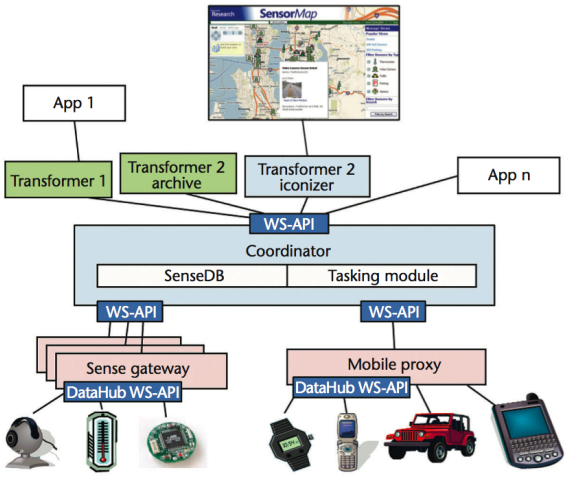


Figura 2.10: SenseWeb: Arquitectura

A arquitectura é constituída por quatro conjuntos lógicos de componentes operacionais: O coordenador (*Coordinator*); Os sensores e respectivos componentes de comunicação com o coordenador (*Sensors, Sensor gateway, Mobile proxy*); Transformadores de dados recolhidos (*Transformers*); E aplicações finais (*Applications*). Estes conjuntos apresentam-se esquematizados na figura 2.10.

O coordenador presta um serviço centralizado, constituindo o ponto de acesso ao sistema quer para as aplicações, quer para fornecedores de sensores. Este é responsável por aceitar pedidos das aplicações, e dar resposta aos mesmos baseado em dados disponíveis em cache, ou gerando uma tarefa de recolha para os sensores indicados. A gestão das tarefas de recolha é feita tendo em conta o grau de ocupação e as capacidades dos sensores, sendo esta indexação de recursos responsabilidade do coordenador.

Dada a heterogeneidade dos sensores implicados no sistema, são necessárias diversas formas de comunicação com os mesmos. Para tornar estes problemas transparentes para o coordenador, o SenseWeb recorre aos *Sensor gateways* e *Mobile proxies*, servido

estes de intermediários entre os sensores e o coordenador. Estes constituem pontos de acesso estáticos à Web, e podem ser suportados num ambiente público, onde podem ser partilhados por diversos sensores, ou podem ser mantidos por cada utilizador que contribua com os seus próprios sensores.

A diferença entre os *Sensor gateways* e as *Mobile proxies* reside no tipo de sensores aos quais dão suporte. Os *Sensor gateways* suportam qualquer tipo de sensores estáticos, sendo responsabilidade das *Mobile proxies* o suporte dos sensores móveis, tornando a mobilidade dos mesmos transparente para as aplicações.

Uma vez recolhidos os dados a partir dos sensores, estes podem necessitar de processamento contextualizado, antes de poderem ser utilizados pelas aplicações. Um exemplo desta situação poderá ser uma aplicação, que necessita de obter o número de pessoas a partir de uma *stream* de vídeo. Nesta situação, é necessário um componente que efectua o processamento das imagens, retornando os dados processados à aplicação. Este é o papel desempenhado pelos *transformers*.

Relativamente à comunicação, nada è referido explicitamente no artigo. No entanto, para tornar transparentes para o coordenador os modos de comunicação de cada sensor, estes são "escondidos"por detrás de *Web Services* suportados pelos *Sensor gateways* e *Mobile proxies*.

2.2.7 COSMOS

O COSMOS [26] consiste numa plataforma de middleware, idealizada como uma camada de uniformização entre redes de sensores genéricas distribuídas geograficamente, e as aplicações de monitorização que operam sobre os dados recolhidos nessas redes. Para fazer uso dos dados recolhidos nestas redes, é necessário às aplicações conhecerem as suas localizações, bem como implementarem APIs específicas para cada rede. O COSMOS pretende uniformizar estas tarefas, tornando transparentes de parte a parte as complexidades de ambas as camadas.

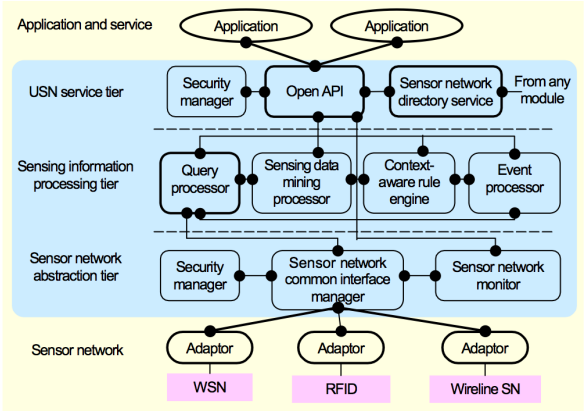


Figura 2.11: Cosmos: Arquitectura

A arquitectura é baseada num serviço centralizado de suporte à plataforma, sendo este divido em três camadas lógicas, como ilustrado na figura figura 2.11: *Sensor network abstraction tier*, que desempenha, através da componente ***Sensor network common interface***, um papel de intermediário entre as redes de sensores e o nível de processamento do COSMOS; ***USN service tier***, responsável pela interacção com as aplicações, mediante a componente *Open API*, que disponibiliza a interface de comunicação para as mesmas através de *Web Services*; ***Sensing information processing tier***, onde ocorre todo o processamento e indexação de *queries* do COSMOS. Este último suporta uma série de funcionalidades acrescidas disponibilizadas às aplicações, no entanto, apenas o *query processor* é obrigatório, uma vez que este modulo é responsável pela execução de *queries* e pelo processamento dos dados provenientes das redes de sensores.

A comunicação das redes de sensores com o COSMOS é efectuada mediante uma política de mensagens definida no *Sensor network common interface*. É mediante esta política de comunicação que a plataforma estabelece ligações com as redes de sensores, bem como as aplicações definem os parâmetros de recolha de dados que necessitam para operar. As redes de sensores ficam responsáveis por implementar os respectivos *Adaptors*, por forma de fazer a ponte entre o modo de comunicação interno à própria rede de sensores e a comunicação com o COSMOS.

Uma vez que as redes de sensores são heterogéneas, existe a preocupação de não sobrecarregar o poder de processamento das redes de sensores mais "fracas", bem como o de aproveitar ao máximo as capacidades das redes mais bem adaptadas. Assim, o COSMOS suporta dois tipo de MTB (*message transport binding*) que devem ser implementados pelos *Adaptors* nas redes de sensores: Uso de XML sobre HTTP e mensagens binárias directamente sobre TCP. A recorrência ao XML garante uma forma versátil de transmissão de dados uma vez que é altamente extensível, e é usado globalmente. No entanto, o processamento de mensagens XML implica a existência de um *parser* que, em sensores com capacidades limitadas, se torna impossível de suportar.

Ao introduzir uma entidade controladora com o intuito de homogeneizar uma generalidade de redes de sensores, é inevitável a introdução de alguma rigidez estrutural nas mesmas. Em redes sustentadas por um servidor centralizado este facto não implicará alterações substanciais, no entanto, para redes totalmente distribuídas torna-se necessária uma forma de eleição de uma entidade reguladora da rede, que suporte este papel.

2.3 Balanço crítico

Os sistemas analisados são uma amostra representativa do estado da arte, onde facilmente se identificam diversas abordagens para resolução da problemática da interacção e comunicação em ambientes móveis ubíquos. Todas elas apresentam vertentes positivas que as tornam plausíveis, bem como pressupostos que nem sempre são passíveis de garantir, criando possíveis pontos de falha ou dificuldades de aplicação em cenários mais abrangentes.

Um ponto que merece principal destaque prende-se com o facto de haver, nos projectos que optam por um modelo cliente/servidor, uma distribuição desequilibrada das responsabilidades entre os dispositivos móveis e os servidores. Nestes, as tomadas de decisão são, sobretudo, provenientes do servidor, funcionando os dispositivos móveis apenas como utensílios de recolha de dados. Assim, as capacidades de processamento dos telemóveis são ignoradas, quando estes se encontram perfeitamente aptos a desempenhar papéis mais autónomos. Este cenário é visível em vários projectos apresentados, tais como Carnet [30], Micro-blog [20] ou Pothole Patrol [18], onde os dispositivos móveis apresentam uma autonomia substancialmente limitada, quando comparados com os papéis desempenhados pelos servidores. Neste trabalho, pretende-se adoptar um modelo de computação distribuída mais pervasivo, procurando que os dispositivos móveis tenham papéis activos, e que sempre que conveniente possam recorrer à infra-estrutura fixa como forma de colmatar limitações em termos de alguns tipos de recursos escassos, tais como capacidade de comunicação, processamento e armazenamento.

Ao dependerem arquitecturalmente de uma só entidade, muitos sistemas, como os identificados anteriormente, estão sujeitos a óbvios problemas de escalabilidade e disponibilidade. Este facto é especialmente crítico nas plataformas de suporte a sistemas de grandes dimensões, como é o caso dos projectos COSMOS [26] e SenseWeb [22, 29]. Esta limitação advém do facto de um servidor ficar responsável por lidar com todos os clientes, constituindo, assim, um ponto central de falha. Para contrapor tal problema, poder-se-ia optar por técnicas de replicação de serviços, no entanto, estas abordagens implicam, inevitavelmente, um consumo acrescido de recursos e uma maior complexidade, decorrentes das necessidades inerentes à coordenação. A importância destas questões, em particular em sistemas com requisitos de pontualidade (tempo real), como aquele que se pretende desenvolver, leva a crer que a escolha de uma arquitectura mais descentralizada, baseada em princípios P2P, poderá responder mais facilmente aos desafios desta área da computação distribuída.

Outra questão pertinente, e que se relaciona com o trabalho a desenvolver, prende-se com os mecanismos de comunicação utilizados. Observa-se que são, tipicamente, exploradas duas alternativas: criação de protocolos próprios ou recorrência a protocolos normalizados. Ao optar pela criação de protocolos próprios (Ex: CTP do Cabernet [17]), consegue-se uma gestão mais controlada dos recursos utilizados. Porém, essas optimizações tendem a limitar a aplicabilidade da solução, pois muitas vezes apenas levam em conta cenários muito concretos. Porém, ao abdicar do recurso a um *middleware* de comunicação, optando pelo desenvolvimento directamente sobre protocolos normalizados, implica que as aplicações tenderão a afrontar o problema da comunicação de um forma *ad hoc* e redutora. A necessidade de reduzir a complexidade da aplicação leva, assim, a não explorar todos os meios de comunicação disponíveis nas plataformas. Uma forma de mitigar o problema, passa por dar maior atenção à forma de organização da informação manipulada. Desta forma, as aplicações fazem uma gestão de alto nível dos recursos, minimizando ao máximo as transmissão de dados redundantes. Esta abordagem é observada no Cartel [24], bem como nas plataformas Haggle [34] e COSMOS [26].

As características de cada projecto analisado no presente capítulo encontram-se sumariadas, respeitando, respectivamente às aplicações relacionadas, tabela 2.1, e arquitecturas e plataformas de comunicação, tabela 2.2.

	Modelo	Modos de comunicação	Tecnologias de comunicação suportadas	Gestão de informação	Tipo de serviço (tempo de comunicação)
Cartel	C/S ou P2P	Servidor: Ligação directa Nos vizinhos: Data muling	WiFi, Bluetooth, etc.	Info propagada para o servidor	Indeterminado (depende de utilizadores para colheita de dados)
Microblog	C/S	Ligação directa ao servidor	WiFi	Info propagada para o servidor	Indeterminado (depende de respostas de utilizadores)
Pothole Patrol	C/S	Ligação directa ao servidor	WiFi ou GPRS	Info propagada para o servidor	Indeterminado (depende de ligações a APs oportunisticas)
Bubblesensing	C/S	Servidor: Ligação directa Nos vizinhos: Broadcast	WiFi ou GPRS/ SMS/ MMS	Info propagada para o servidor	Tarefas despoletadas por telemóveis: Tempo-real (derivado da proximidade de utilizadores)
Mobeyes	P2P	Data muling	VANET	Mantida pelos veículos, até ser colhida	Indeterminado (derivado de data muling)
Mobile Chedar	P2P	*System multicasting*	Bluetooth	Mantida pelos nós Chedar	Tempo-real (derivado da proximidade implicada pelo bluetooth)
LEGENDA:	C/S: Cliente/Servidor; P2P: *Peer To Peer*; VANET: *Vehicular Ad Hoc Networks* (802.11p) [1,25] AP: *Access Point*				

Tabela 2.1: Organização sumária de aplicações relacionadas

	Suporta aplicações a operar sobre o modelo	**Modos de comunicação**	**Tecnologias de comunicação suportadas**
DEEDS	C/S ou P2P	Publish/Subscribe por canais	Independente das tecnologias de comunicação
Cafnet	C/S ou P2P	Servidor: Ligação directa Nos vizinhos: Data muling	WiFi, Bluetooth, etc.
Cabernet	C/S	Ligação directa ao servidor	WiFi
Exploiting our Comput.	P2P	Independente das tecnologias de comunicação	Independente das tecnologias de comunicação
Haggle	C/S ou P2P	Independente das tecnologias de comunicação	Independente das tecnologias de comunicação
SenseWeb	C/S	Independente das tecnologias de comunicação	Independente das tecnologias de comunicação
Cosmos	C/S	Ligação directa ao servidor	Independente das tecnologias de comunicação
LEGENDA:	C/S: Cliente/Servidor; P2P: *Peer To Peer*;		

Tabela 2.2: Organização sumária de arquitecturas e plataformas de comunicação

3

Especificação e Desenho

3.1 Problema

O desafio abordado nesta dissertação consiste em conseguir que num sistema distribuído móvel de larga escala e disperso geograficamente, os nós consigam encontrar potenciais parceiros, seleccionados com base em critérios geográficos e temporais.

Os potenciais parceiros de um nó móvel são aqueles que num mesmo local e num mesmo instante têm interesses em comum com o nó em questão. Ou seja, são aqueles que segundo a lógica das aplicações reúnem certas condições para que devam interagir com o nó em questão.

3.1.1 Condicionantes/Premissas

Pretende-se resolver esta questão com base num conjunto de restrições que reflictam as condições do ambiente de operação. Neste ambiente, os nós móveis conhecem em qualquer instante a sua localização geográfica, bem como o destino e os percursos das suas trajectórias.

Os nós operam de forma isolada, sem existência quer de referências pré-acordadas entre si, quer de servidores centralizados. Estes têm a capacidade de comunicar através de um substrato *content-based routing*, segundo um modelo *publish/subscribe*, com filtragem de eventos baseada no conteúdo.

Os filtros são genéricos, e correspondem a zonas geográficas usadas para determinar potenciais nós parceiros. No entanto, estes podem impor outro tipo de restrições

baseadas, por exemplo, em critérios temporais. É necessário ter em consideração que a actualização frequente de áreas de filtragem é desencorajada, pois sabe-se que esta consiste numa operação demasiado dispendiosa para ser executada com uma frequência elevada.

As áreas de filtragem podem tomar qualquer forma desejada, corresponder a áreas disjuntas, e ser centradas em qualquer ponto geográfico independentemente da localização do nó que as define. Assim, estas podem representar a zona circundante de um nó (figura 3.1 - A), bem como uma trajectória (figura 3.1 - B), não estando dependentes da posição do nó, nem sujeitas a limitações relativas ao número de áreas definidas (figura 3.1 - C). Desta forma, as aplicações podem definir os critérios de selecção de potenciais parceiros que melhor respondam às suas necessidades, usando o substrato *content-based routing* para eliminar parceiros indesejados.

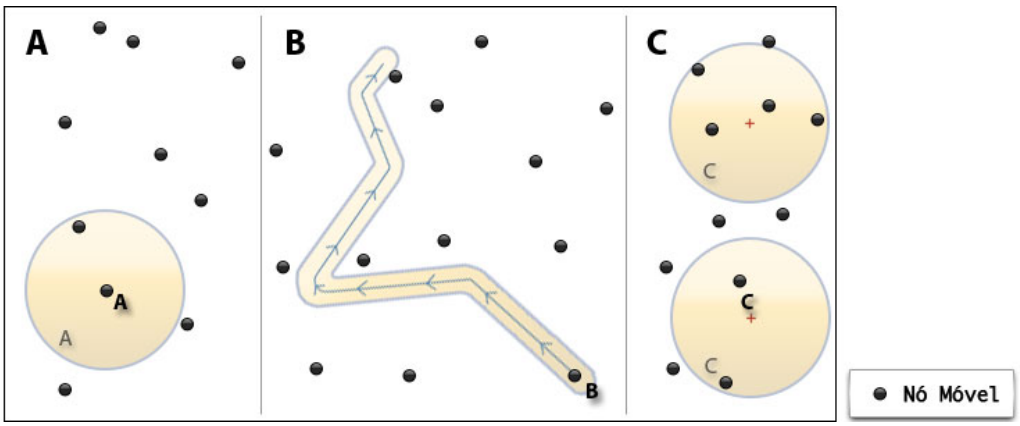


Figura 3.1: Possíveis representações de áreas de filtragem

3.1.2 Objectivos

Qualquer solução para o problema apresentado deve ser eficaz, na medida em que deve conseguir seleccionar o maior número de potenciais parceiros num espaço de tempo reduzido. Procura-se também que esta seja eficiente, devendo minimizar os custos de comunicação e a complexidade de operação, por forma a possibilitar a operação em dispositivos móveis.

As soluções devem ser igualmente dinâmicas e escaláveis, devendo lidar bem com a frequente entrada e saída de participantes no sistema, mantendo as garantias de eficiência, independentemente do crescimento do sistema. Estas garantias são em grande parte relegadas para o substrato *publish/subscribe* usado para suporte à comunicação. Este substrato é parte integrante da plataforma que sustenta as soluções, descrita no capítulo seguinte.

3.2 Cenários

No contexto aplicacional, os nós móveis estão tipicamente associadas a indivíduos que actuam enquanto utilizadores. Estes utilizadores apresentam um conjunto de interesses que, à luz do problema em questão, são definidos por critérios geográficos e temporais. Significa isto que, os utilizadores seleccionam os potenciais parceiros através da definição de zonas de geográficas com uma duração associada.

No entanto, é necessário ter em consideração que a mobilidade dos indivíduos terá influência na definição destas zonas, uma vez que esta não deve ser limitada pela definição de áreas de selecção, ou seja, a definição de áreas geográficas deve ter em conta o facto de que o indivíduo não deve ser forçado a permanecer estático durante o período em que estas estão activas.

Como exemplo deste facto, considere-se o cenário em que um indivíduo, a cargo de uma distribuidora, pretende seleccionar todos os clientes localizados ao longo do seu itinerário, que devem receber encomendas. Para tal, a zona geográfica de selecção de clientes será baseada no itinerário do condutor, ou seja, a mobilidade do condutor exerce uma influência directa na definição da área de selecção de clientes.

Desta forma, é necessário analisar cada um dos casos de selecção de potenciais parceiros, com base nos diferentes tipos de áreas de interesse possíveis, sendo estes ditados pela mobilidade dos intervenientes.

3.2.1 Cenário 1 - Estacionário / Estacionário

O primeiro cenário contempla o caso em que as áreas de interesse dos utilizadores recaem sobre zonas concretas. Ou seja, os utilizadores estão interessados em encontrar potenciais parceiros com base em zonas particulares, e não nas zonas abrangidas pela mobilidade desses mesmos potenciais parceiros.

No contexto do *car pooling*, os potenciais parceiros são naturalmente quem necessita de uma boleia e quem a oferece. Para que dois utilizadores, neste papéis, estejam em condições de negociar uma boleia, é necessário que estes estejam próximos, e que se pretendam mover para destinos igualmente próximos, possivelmente distantes dos iniciais. Assim, as áreas de interesse de ambos os utilizadores recaem sobre as zonas circundantes tanto da sua localização corrente, como da localização do ponto de destino para onde se pretendem mover, sendo estas definidas por um círculo com um raio que reflecte a tolerância que os utilizadores estão dispostos a aceitar para estipular uma boleia.

No caso de se verificar que tanto a origem como o destino de um condutor e de um utilizador que necessite de boleia se intersectem, tal como ilustrado na figura 3.2, então estes dois utilizadores estão em condições de negociar uma boleia.

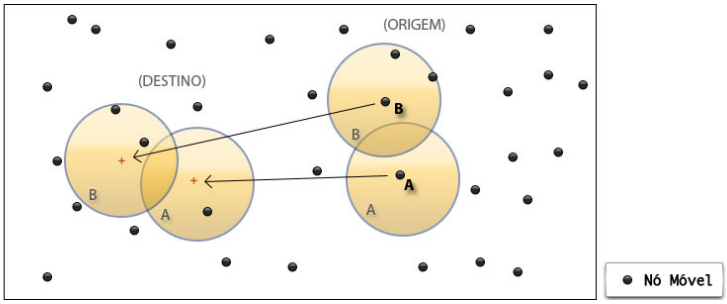


Figura 3.2: Cenário 1

3.2.2 Cenário 2 - Estacionário / Em Movimento

Este cenário comporta o caso em que a mobilidade é considerada influente na definição de áreas de selecção de potenciais parceiros por parte de um dos intervenientes. Esta mobilidade tanto pode ser associada ao próprio utilizador como aos potenciais parceiros que este pretende encontrar. Assim, a interacção ocorre entre um utilizador que, tal como no cenário anterior, define as suas áreas de interesse com base em zonas concretas e bem delimitadas, e outro que contempla a mobilidade na definição das suas área de interesse.

Este cenário comporta o caso em que a mobilidade de um dos intervenientes é considerada influente na definição de áreas de selecção de potenciais parceiros, ou seja, a interacção ocorre entre um utilizador que, tal como no cenário anterior, define as suas áreas de interesse com base em zonas concretas e bem delimitadas, e outro que contempla a mobilidade na definição das suas áreas de interesse.

No contexto do *car pooling*, será o condutor que irá basear as suas áreas de selecção no movimento do indivíduo, pois assume-se que quando um indivíduo inicia uma viagem, este sabe a trajectória que vai tomar, ainda que de modo grosseiro, podendo esta ser calculada a partir de um qualquer *software* de navegação instalado no dispositivo móvel.

Assim, o utilizador que solicita uma boleia mantém o comportamento observado no cenário anterior, definindo duas zonas de selecção circundantes da sua localização e da localização do destino para onde se pretende deslocar, ambas baseadas num raio de tolerância. Por sua vez, o condutor pode usar a sua trajectória para definir a zona de

filtragem como sendo a área circundante dos troços que vai seguir, tal como ilustrado na figura 3.3.

Caso a área geográfica definida pelo condutor intersecte ambas as zonas de selecção do requisitante e, caso o condutor de desloque no sentido das necessidades do pedido, isto é, desde a localização do requisitante até á posição de destino do seu pedido, então estes dois utilizadores estarão em condições de estabelecer ou negociar uma boleia.

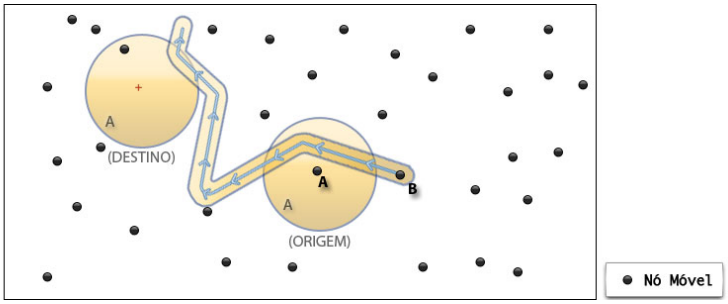


Figura 3.3: Cenário 2

3.2.3 Cenário 3 - Em Movimento / Em Movimento

Por último, considera-se o cenário que comporta a interacção entre dois utilizadores que definem os seus interesses baseados na mobilidade, isto é, ambos pretendem seleccionar potenciais parceiros com base em zonas de interesse que reflictam a mobilidade destes, e a do próprio utilizador.

No contexto do *car pooling*, o condutor pode ser visto de forma idêntica à apresentada no cenário anterior, definindo a sua área de interesse com base no seu itinerário. Por sua vez, o utilizador que solicita a boleia passa a considerar a sua própria mobilidade para definir as suas zonas de interesse. No entanto, ao contrário do condutor, não se assume que o indivíduo que solicita uma boleia conhece de antemão a trajectória da sua mobilidade, enquanto aguarda por resposta ao seu pedido. A razão para tal deve-se ao facto de, por um lado os indivíduos que solicitam boleias não estarem limitados a regras de mobilidade (regras de trânsito), e por outro, não conhecem à partida o tempo que vão estar à espera de resposta, pelo que não têm forma de planear a sua mobilidade de forma fundamentada.

Assim, a zona circundante do ponto de destino do pedido continua a ser definida por um círculo centrado neste ponto com um raio de tolerância associado, ao passo que a zona circundante da localização do utilizador passa a ser definida à custa de círculos, centrados nesta localização, com um raio de tolerância associado, que vão sendo actualizados periodicamente. A figura 3.4 ilustra o cenário descrito.

Caso a área geográfica definida pelo condutor intersecte ambas as zonas de selecção do requisitante e, caso o condutor de desloque no sentido das necessidades do pedido, isto é, desde a localização do requisitante até à posição de destino do seu pedido, então estes dois utilizadores estarão em condições de estabelecer uma boleia.

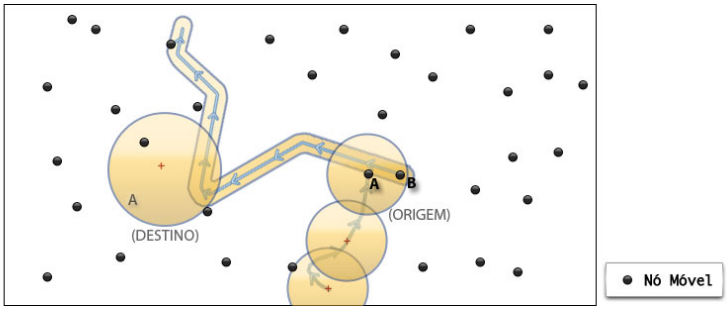


Figura 3.4: Cenário 3

3.3 Desenho da Solução

Uma vez identificados e descritos tanto o problema abordado como as ferramentas disponíveis para o resolver, descrevem-se nesta secção os traços gerais que sustentam cada uma das soluções propostas. Estas são apresentadas em pormenor no capítulo seguinte, referente à implementação do protótipo.

3.3.1 Arquitectura da Solução

Até este ponto, a discussão do problema tem sido centrada unicamente nos nós móveis. No entanto, a solução pretendida, sendo descentralizada, não é puramente *P2P*, envolvendo outras componentes organizadas numa infra-estrutura fixa.

Esta secção tem como objectivo descrever a arquitectura subjacente às soluções propostas, cuja composição tem por base dois níveis de abstracção: **Nível aplicacional** e **Nível de sistema**.

Estes estão organizados em camadas sobrepostas, tal como ilustrado na figura 3.5. O nível de aplicacional corresponde à camada de abstracção que alberga a lógica das aplicações, ao passo que o nível de sistema corresponde à camada de abstracção que alberga a lógica da plataforma de comunicação.

Figura 3.5: Camadas de abstracção da Arquitectura das Soluções

3.3.1.1 Nível Aplicacional

Neste nível são definidas duas componentes lógicas idealizadas para operar sobre as entidades móveis e fixas, como ilustrado na figura 3.6. A razão para tal é justificada pela escassez de recursos das entidades móveis quando comparadas com as entidades fixas. Desta forma, as aplicações delegam a complexidade do processamento dos dados contextuais da aplicação para as entidades fixas.

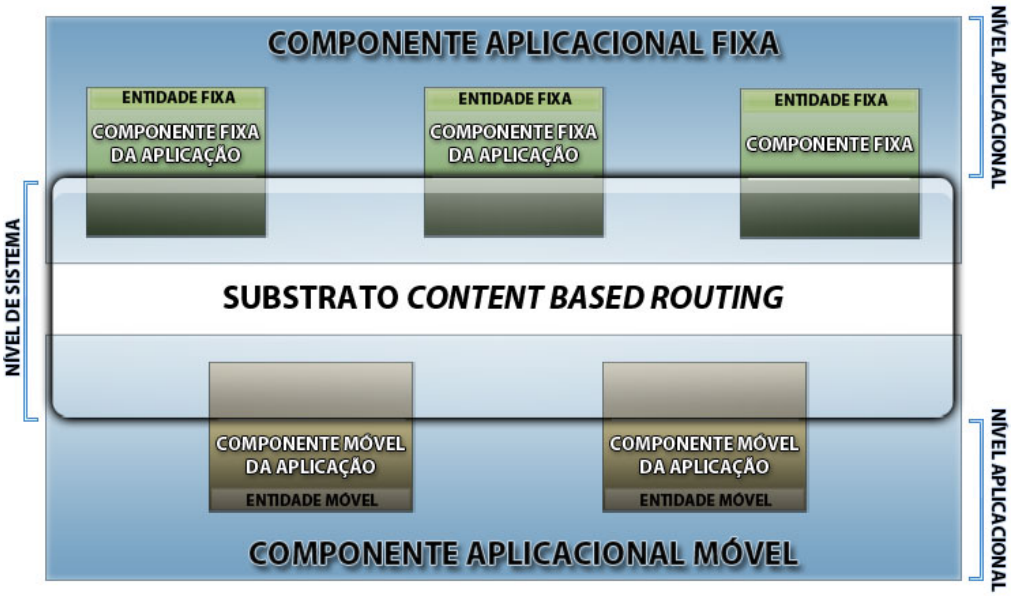


Figura 3.6: Arquitectura da solução - Nível aplicacional

- **Componente Móvel**

  A componente móvel consiste num módulo aplicacional a correr nos dispositivos móveis dos utilizadores, e o seu objectivo consiste em providenciar o meio através do qual estes interagem com as aplicações.

  A partir deste módulo, é recolhida a informação contextual usada para encontrar os potenciais parceiros do utilizador, mediante os critérios impostos pela lógica das aplicações, ou seja, a componente móvel é responsável pela recolha dos dados, através de interacção com o utilizador, que alimentam as aplicações. No contexto da presente dissertação, as aplicações correspondem às diferentes soluções que serão descritas na secção 3.3.2.

  A informação contextual consiste, entre outros dados, na localização geográfica do utilizador, pelo que a componente móvel tem a capacidade de conhecer a qualquer altura a sua localização geográfica, através por exemplo de um sensor GPS.

- **Componente Fixa**

  As componentes fixas, por operarem sobre entidades fixas, gozam de boas capacidades de processamento e armazenamento, estando bem conectadas e sem limitações de energia. Estas estão distribuídas geograficamente, conhecendo bem as suas localizações e seu objectivo consiste em fornecer facilidade de comunicação, capacidade de processamento e colaboração entre as componentes móveis.

  Desta forma, recai sobre as componentes fixas a responsabilidade de garantir o suporte computacional para processamento dos dados contextuais recolhidos pelas componentes móveis.

3.3.1.2 Nível de Sistema

As responsabilidades do nível de sistema podem ser divididas em duas partes distintas. Por um lado, é responsável por suportar os serviços de comunicação, pelo que é a este nível que é implementado o substrato *content based routing*. Por outro lado, é responsável por executar código aplicacional, ou seja, existe uma relação estreita entre a camada aplicacional e a camada de sistema, uma vez que as entidades definidas na camada aplicacional são definidas com base nas entidades que compõem o nível de sistema.

A figura 3.7 ilustra as componentes que compõem o nível de sistema. Comparando esta ilustração com a figura 3.6, podemos observar a relação mencionada entre as entidades lógicas do nível aplicacional e as do nível de sistema, pois ambas partilham as mesmas entidades (móveis ou fixas).

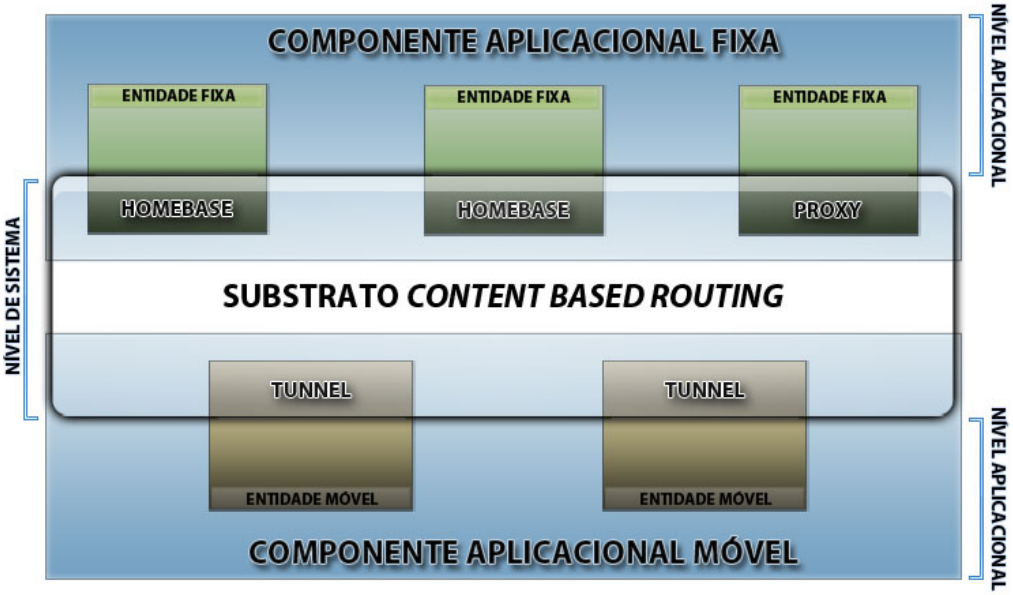


Figura 3.7: Arquitectura da solução - Nível de sistema

As entidades do nível de sistema são as seguintes:

- ***Homebase***

  A *homebase* consiste numa entidade fixa, que aloja serviços de sistema para uma entidade móvel particular. Estes serviços consistem tanto em assegurar a conectividade da entidade móvel à infra-estrutura fixa, como oferecer capacidade computacional para colmatar a necessidade de recursos das entidades móveis.

  As *homebases* funcionam numa associação de 1 para 1 com as entidades móveis, no sentido em que cada *homebase* tem apenas uma entidade móvel associada, e vice-versa. A camada aplicacional tira partido desta associação para definir as componentes fixas das aplicações, pois estas partilham da mesma lógica de associação no sentido em que cada componente fixa implica uma componente móvel a si associada e vice-versa.

- ***Proxy***

  Os *proxies* consistem em entidades fixas, que desempenham serviços de *roaming* a nível do substrato de comunicação. No entanto, além destas funções, estes poderão alojar funcionalidades que servem de benefício para todas as instâncias da aplicação móvel.

  Estas funcionalidades são definidas a nível aplicacional que, tal como acontece com as *homebases*, tiram partido do facto de os *proxies* representarem entidades bem definidas no nível de sistema. Assim, na camada aplicacional, os *proxies* desempenham o papel de *brokers* para as instâncias da aplicação móvel. Estes são nós que não são pré-definidos, e são seleccionados dinamicamente mediante critérios geográficos.

- ***Tunnel***

  O *Tunnel* representa um componente que opera nas entidades móveis, e que permite que estas consigam comunicar com as entidades fixas (*homebases* e *proxies*). Este componente encontra-se descrito mais detalhadamente na secção 4.2.2.

3.3.2 Soluções Propostas

O desenho das abordagens encontra-se orientado ao contexto do *car pooling*, e pretende servir como base introdutória para as abordagens implementadas no capítulo seguinte.

Para resolver o problema do encontro de entidades com base no modelo de comunicação descrito na secção 3.1.1, é necessário ter dois factores presentes. Em primeiro lugar, toda a comunicação é feita por meio de mensagens, em segundo, é possível definir áreas geográficas de filtragem que restringem a recepção dessas mensagens.

Estes dois princípios constituem a base sobre a qual são desenvolvidas todas as soluções propostas nesta secção.

3.3.2.1 Solução para Cenário 1

Nesta secção descreve-se a solução referente ao cenário 1 (secção 3.2.1). Neste, observámos que os utilizadores estão interessados em encontrar potenciais parceiros com base em zonas geográficas bem definidas, ou seja, a mobilidade dos utilizadores é irrelevante para o processo, bastando conhecer apenas a sua localização corrente e a posição de destino a que o seu pedido ou a sua oferta correspondem.

Desta forma, os pedidos e as ofertas apresentam um tipo comum, pois são constituídas por dados semelhantes que diferem apenas no contexto em que são usados, isto é, no caso de um pedido a posição corrente é relativa ao requisitante e o destino à posição para onde o requisitante se pretende deslocar. No caso de uma oferta, a posição corrente é relativa ao condutor e o destino à posição para onde este se vai deslocar.

Tendo em conta que as entidades constituintes da infra-estrutura fixa se encontram distribuídas geograficamente, e conhecem bem as suas localizações, podemos tirar partido destas para desenvolver a solução para este cenário, em vez de responsabilizar unicamente as componentes móveis pelos seus próprios pedidos ou ofertas.

Desta forma, se os *proxies* definirem uma área geográfica de filtragem correspondente a um círculo centrado na sua localização, com um raio que reflicta a tolerância a que estes estão dispostos a aceitar mensagens, estes passam a aceitar e a armazenar todos os pedidos e ofertas cujos destinos se localizem no interior do referido círculo de tolerância. Assim, os *proxies* servem de ponto de encontro entre os pedidos e as ofertas.

Porém, pode ocorrer a situação em que o destino de um pedido ou de uma oferta contidos numa mensagem, não se intersectam com nenhum círculo de tolerância de

nenhum *proxy*. Significa isto que não existe qualquer entidade que garanta a persistência do pedido ou da oferta em questão. Para solucionar este problema, são propostas duas alternativas:

- **Aproximação baseada nas entidades fixas**

  Para resolver a situação descrita, podemos tirar partido do facto de as componentes móveis terem sempre uma referência para uma entidade fixa que alberga a sua componente fixa. Assim, caso se detecte a não existência de um *proxy* que se responsabilize por um pedido ou oferta, sendo esta situação detectada com base numa política de *timeout* de mensagens, podemos delegar essa tarefa para esta entidade fixa que define um círculo de filtragem centrado na posição de destino do pedido ou oferta, com um raio de tolerância parametrizado. Significa isto que, na inexistência de um *proxy* que se responsabilize por uma mensagem, a entidade fixa conhecida da componente móvel assume essa responsabilidade. A figura 3.8 ilustra a aproximação descrita.

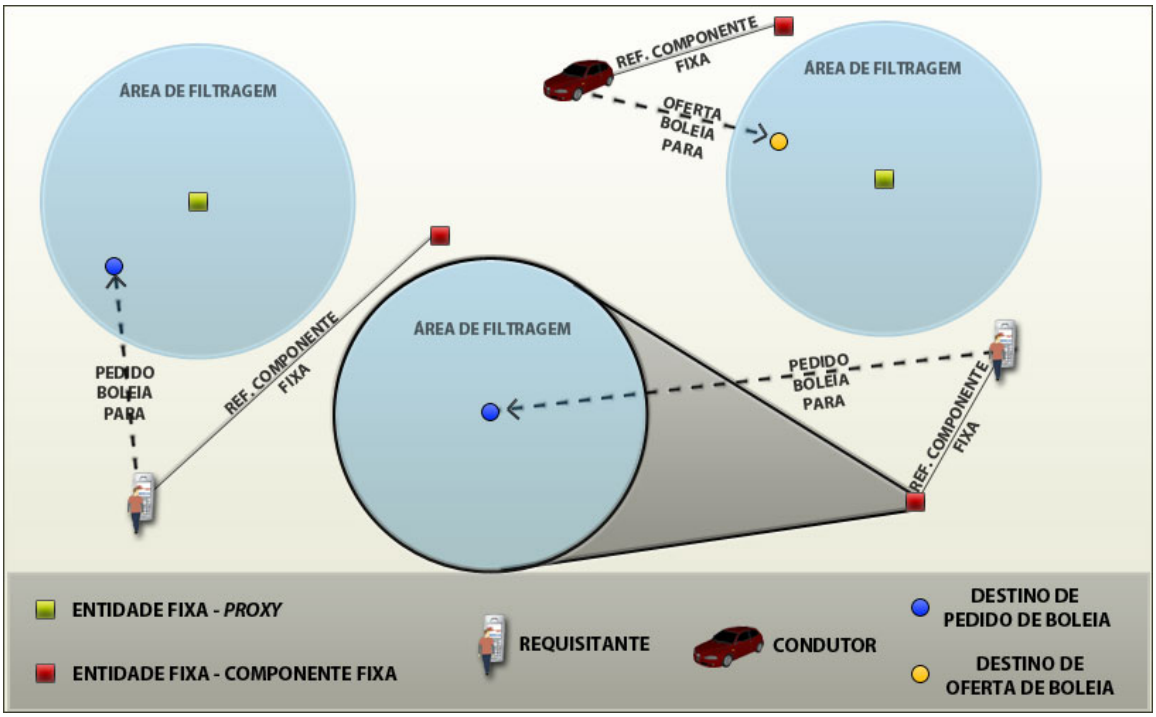


Figura 3.8: Solução para cenário 1 - Aproximação baseada nas entidades fixas

- **Aproximação baseada nas mensagens**

  A segunda alternativa consiste no transporte da complexidade do processo de responsabilização dos *proxies* para as mensagens, isto é, em vez de serem os *proxies* a definir um círculo de filtragem, passam a ser as mensagens a transportar essa informação. Assim, as mensagens passam a ser compostas pelos dados do

utilizador da componente móvel, da sua posição corrente, do destino da oferta ou pedido em questão e de um raio de tolerância. Os *proxies* por sua vez, passam a definir os seus filtros baseados unicamente na sua posição geográfica, ficando responsáveis por todos os pedidos, transportados em mensagens, cujos círculos de tolerância contenham esta posição.

Se ao publicar uma mensagem de pedido ou oferta de boleia, a componente móvel não obtiver resposta por parte de nenhum *proxy*, então esta passa a publicar a mesma mensagem com um valor de raio de tolerância maior. O processo é repetido até que a componente móvel receba resposta de um *proxy* a garantir a persistência do pedido ou oferta, ou até que seja atingido um limite de incremento de raio de tolerância. A figura figura 3.9 ilustra a aproximação descrita.

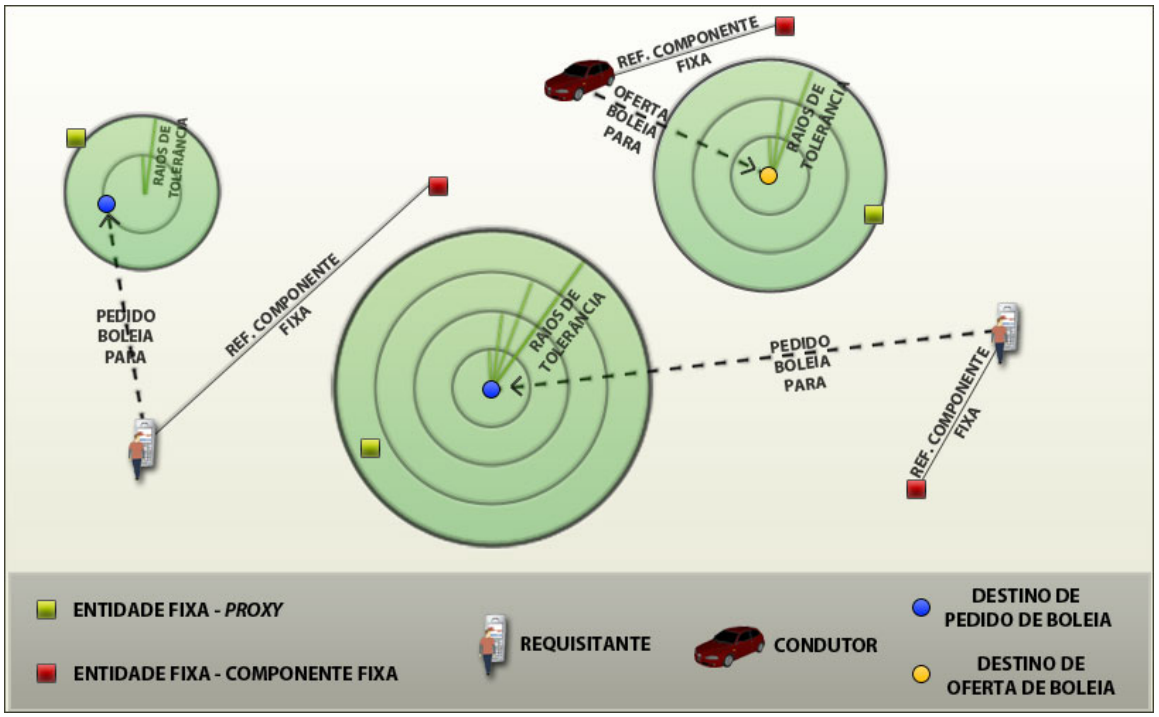


Figura 3.9: Solução para cenário 1 - Aproximação baseada nas mensagens

Com base na mecânica descrita em ambas as aproximações, o processo de determinação de condutores em condições de oferecer uma boleia a um requisitante fica simplificado, pois o universo de ofertas possíveis fica limitado às ofertas que estão a cargo do mesmo *proxy* que se responsabiliza por um pedido. Assim, sempre que um *proxy* receba um pedido, efectua uma pre-selecção de ofertas, para eliminar todas aquelas cujos condutores se encontrem demasiado distantes do requisitante, e responde com a lista de ofertas remanescentes. O requisitante por sua vez, encarrega-se de fazer a escolha da oferta mais promissora e de contactar o respectivo condutor.

3.3.2.2 Solução para Cenário 2

Ao propor uma solução relativa ao cenário 2 (secção 3.2.2), é necessário ter em consideração a diferença de comportamentos existente entre os utilizadores que oferecem boleias e os que as solicitam. Esta diferença ocorre pela necessidade de contemplar a mobilidade dos condutores no processo de estabelecimento de boleias. Tal implica que a solução apresentada para o cenário 1 é inadequada para o presente cenário, pois as mensagens de pedido e de oferta deixam de ser de um tipo comum, visto a posição do condutor se mover.

Desta forma, os *proxies* deixam de poder tomar responsabilidade pelas mensagens da mesma forma que o faziam na solução anterior, pelo que se torna necessário encontrar outra forma de garantir a persistência das ofertas e dos pedidos de boleias dos utilizadores. Para tal, pode ser tirado partido do facto de as componentes móveis conhecerem a qualquer instante as referências para as componentes fixas às quais estão associados.

Antes de determinar as responsabilidades atribuídas a cada uma das componentes envolvidas no processo, é necessário analisar as condições perante as quais um condutor e um requisitante se encontram em posição de negociar uma boleia. Assim, assumindo que os condutores conhecem de antemão os itinerários que vão percorrer, um requisitante e um condutor encontram-se em condições de negociar uma boleia sempre que o itinerário de um condutor obedeça a duas condições. Por uma lado, tem que passar perto tanto da localização do requisitante como do destino do seu pedido, e por outro, tem que ocorrer de acordo com as necessidades do pedido, ou seja, o condutor tem que se deslocar de forma a que passe primeiro perto do requisitante e só depois do destino do pedido deste. A figura 3.10 ilustra o cenário em que um condutor e um requisitante estão em condições de acordar uma boleia

Desta forma, a solução para o presente cenário desenrola-se da seguinte forma: Quando um utilizador, a actuar enquanto requisitante, pretender efectuar um pedido de boleia, este encaminha os dados relativos ao pedido para a componente fixa a que está associado. Estes consistem na sua posição geográfica, no destino do pedido, num período de duração para o pedido e de um raio de tolerância usado para determinar se o grau de proximidade de uma trajectória de um condutor é aceitável ou não. Por sua vez, a componente fixa define uma área de filtragem correspondente a dois círculos, centrados na localização do utilizador e no destino do pedido, ambos com um raio igual ao raio de tolerância.

Para efectuar uma oferta de boleia, o condutor publica periodicamente mensagens contendo a sua posição corrente, e os troços que constituem o trajecto que falta percor-

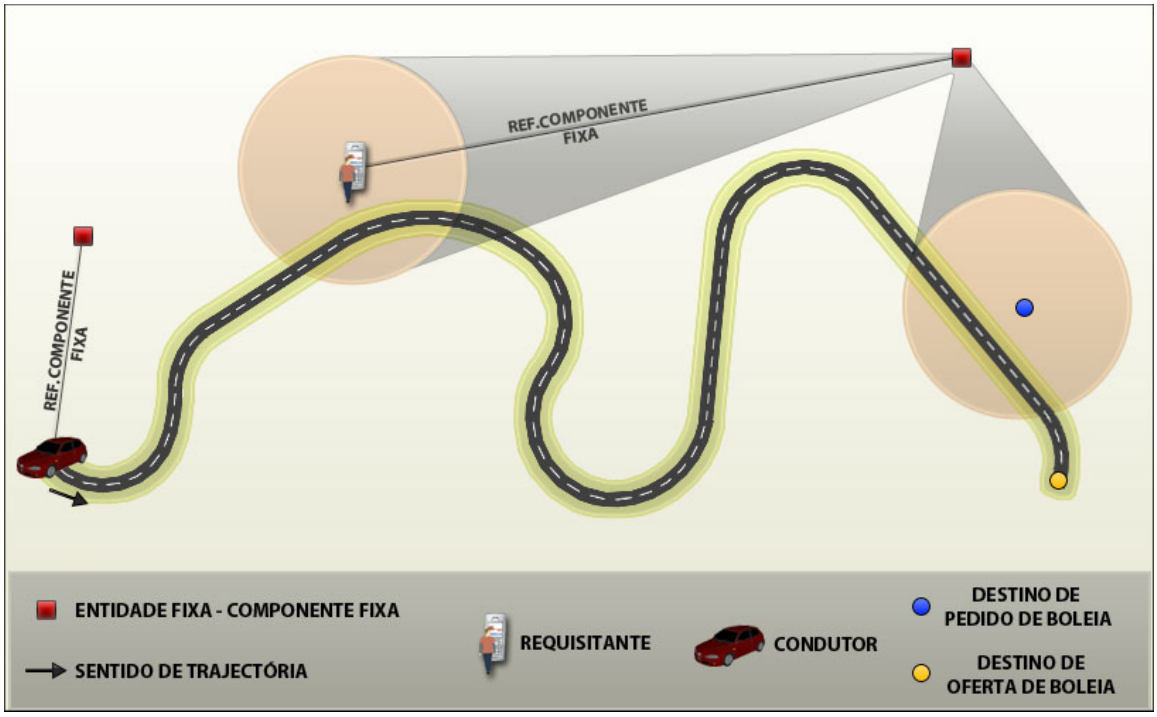


Figura 3.10: Solução para cenário 2

rer do seu itinerário, ou seja, os troços que já foram percorridos vão sendo descartados. Se uma boleia intersectar ambos os círculos que constituem a zona de filtragem de uma componente fixa, e se a deslocação do condutor se der de acordo com as necessidades do pedido, então a mensagem é aceite pela componente fixa, que a encaminha para o requisitante a que está associada. Este por sua vez, encarrega-se de fazer a escolha da oferta mais promissora de entre todas as que recebe e de contactar o respectivo condutor.

É importante referir que existe uma alternativa a esta solução, pois poderiam ser as componentes fixas, associadas aos condutores, a definir áreas de filtragem correspondentes às zonas circundantes das trajectórias dos condutores. Nesta situação, seriam os requisitantes que publicariam mensagens com os dados relativos ao pedido, sendo que o processo de filtragem se mantinha idêntico. Porém, esta alternativa não foi escolhida, uma vez que se assume que o período de duração dos pedidos de boleia é superior face ao tempo que leva em média um condutor a percorrer uma trajectória. Desta forma, minimiza-se a necessidade de actualização de filtros que, como já foi referido, trata-se de uma operação dispendiosa.

3.3.2.3 Solução para Cenário 3

Nesta secção descreve-se a solução referente ao cenário 3 (secção 3.2.3), em que existe a total mobilidade os utilizadores. No entanto, ao contrário do que acontece com os condutores, não podemos assumir que os utilizadores que solicitam boleias conhecem em antemão os caminhos que percorrem, durante o período em que aguardam pelo resultado da pesquisa por boleias viáveis.

Significa isto que a solução proposta para o cenário 2 não pode ser aplicada nesta situação, pois nesta solução, as áreas de filtragem são definidas à custa quer da localização do requisitante quer do ponto de destino para onde este se pretende mover. No presente cenário, isso implicaria uma necessidade de actualização constante das áreas de filtragem, o que a este ponto se sabe ser uma operação dispendiosa.

No entanto, podemos ponderar a alternativa apontada na descrição da solução anterior, que remete a definição áreas de filtragem de mensagens para as componentes fixas associadas aos condutores. Desta forma, a solução para o presente cenário baseia-se na definição de áreas de filtragem assentes sobre os itinerários dos condutores. Assim, quando um condutor pretender oferecer uma boleia, este encaminha para a componente fixa a que está associado, uma mensagem contendo a lista de pontos interpoladores do seu itinerário. Esta por sua vez, define uma área de filtragem correspondente à zona circundante a todos troços que compõem a trajectória do condutor.

Por outro lado, um requisitante para efectuar um pedido de boleia, publica periodicamente mensagens contendo os dados relativos ao pedido. Estes consistem na localização corrente do utilizador que actua enquanto requisitante, na posição de destino do pedido, no período de duração do pedido e num raio de tolerância, usado para determinar as boleias compatíveis. Porém, dada a mobilidade do utilizador, as mensagens relativas a um mesmo pedido vão sofrendo alterações, uma vez que a posição corrente do utilizador vai sendo alterada, bem como o tempo de duração do pedido, que é diminuído de acordo com a periodicidade de publicação.

O processo de determinação de viabilidade de boleias para a presente solução é idêntico ao que é descrito na solução para o cenário 2. Porém, uma vez que as áreas de filtragem são baseadas nos itinerários completos do condutor, as componentes fixas associadas ao condutor vão receber todas as mensagens relativas a pedidos que cumpram as regras de filtragem já descritas, independentemente da posição do condutor. Assim, compete à componente fixa associada ao condutor, efectuar um processo de filtragem adicional que descarte todos os pedidos recebidos cuja posição do requisitante se encontre perto de um troço pelo qual o condutor já passou.

Ao receber um pedido que se encontre em condições de ser satisfeito pela oferta do condutor, a componente fixa associada a este responde ao requisitante a notificar a viabilidade da boleia. Este por sua vez, encarrega-se de fazer a escolha da oferta mais promissora, de entre todas as ofertas que recebe, e de contactar o respectivo condutor com o objectivo de negociar uma boleia.

A figura 3.11 exemplifica uma situação que contempla a mobilidade do utilizador que solicita boleias. Na primeira fase - A, o utilizador não se encontra em situação de ser satisfeito pela boleia oferecida, pelo que as suas mensagens publicadas não serão aceites pela componente fixa responsável pela zona de filtragem do condutor. No entanto, ao se movimentar até à posição ilustrada em B, as suas mensagens passam a ser aceites, e uma vez que o condutor ainda não passou pela posição do requisitante, estes estão em condições de negociar uma boleia.

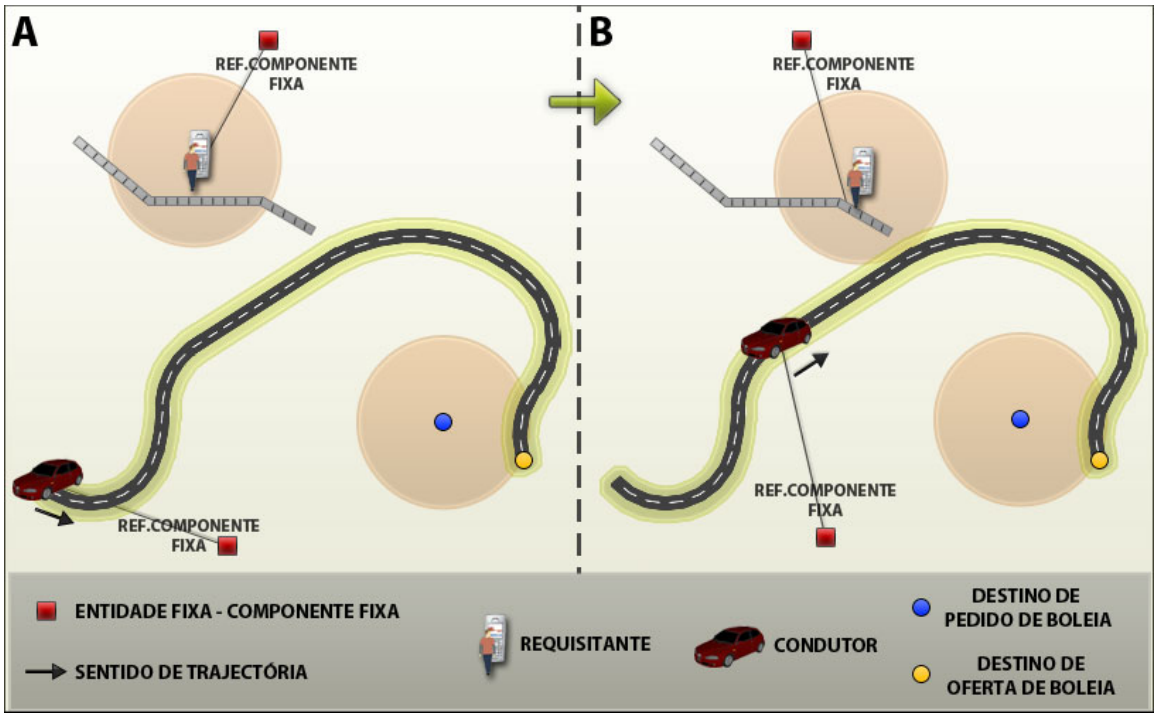


Figura 3.11: Solução para cenário 3

4

Protótipo

4.1 Enquadramento

O presente capítulo é referente ao processo de desenvolvimento das soluções propostas na secção anterior. Neste é apresentada a plataforma que serviu de suporte ao processo de implementação e teste das respectivas soluções.

Esta plataforma, denominada MEEDS [31], consiste numa plataforma de comunicação capaz de oferecer suporte a computação móvel e ubíqua.

Relativamente à organização da presente secção, primeiramente é apresentada a descrição da arquitectura da plataforma MEEDS, bem como do modelo de programação usado na implementação das soluções. Posteriormente apresenta-se, nesta ordem, a descrição quer do caso de estudo que contextualiza as soluções, quer do processo de implementação das mesmas, tendo esta última como base o modelo disponibilizado pela plataforma e as premissas apresentadas na contextualização do caso de estudo.

4.2 MEEDS

MEEDS [31] consiste numa plataforma de disseminação de eventos extensível e distribuída com suporte para mobilidade. Esta é exposta segundo um modelo *pusblish/-subscribe* com filtragem baseada no conteúdo. A comunicação é feita sobre canais de eventos activos [16], que representam um fluxo de mensagens com uma relação lógica entre si, como por exemplo os eventos de uma aplicação ou de um grupo de aplicações

relacionadas. Esta lógica está normalmente associada à semântica dos conteúdos das mensagens ou ao suporte de uma determinada qualidade de serviço. Estes canais permitem igualmente a capacidade de fazer *feedback* a eventos recebidos, ou seja, enviar um evento que serve de resposta a um evento concreto que tenha sido recebido. Desta forma, MEEDS segue aquilo que se poderia chamar um modelo *publish/subscribe/feedback*.

A MEEDS é definida como uma extensão à plataforma fixa FEEDS (sendo esta reimplementação de DEEDS [16]), que implementa o serviço de disseminação de eventos restritos ao nível da infra-estrutura fixa. Esta extensão é conseguida a partir da definição de novos componentes que se enquadram na arquitectura de FEEDS enquanto nós clientes. A figura 4.1 ilustra a relação descrita entre MEEDS e FEEDS.

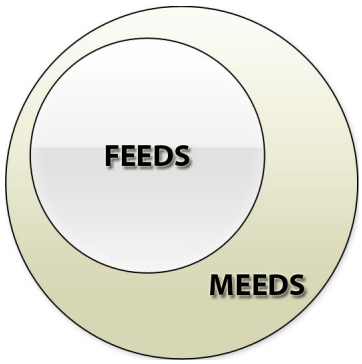


Figura 4.1: Modelo lógico da plataforma MEEDS

4.2.1 Arquitectura FEEDS

A arquitectura da plataforma FEEDS consiste numa rede *overlay* hierarquizada em 3 camadas. Esta rede é composta por 3 tipos de nós, sendo estes **Servidores primários**, **Servidores secundários** e **Nós clientes**. Tanto os servidores primários como os servidores secundários são caracterizados por uma grande disponibilidade de recursos, sendo entidades capazes de processar grandes volumes de dados rapidamente. Os nós clientes podem ser encarados como computadores pessoais, que dispõem de uma menor capacidade de recursos face a ambos os tipos de servidores, e que albergam as aplicações que operam sobre a plataforma.

Ao dispor a organização mediante camadas, como ilustrado na figura 4.2, FEEDS pretende oferecer tanto uma capacidade de adaptabilidade como assegurar a capacidade de escala.

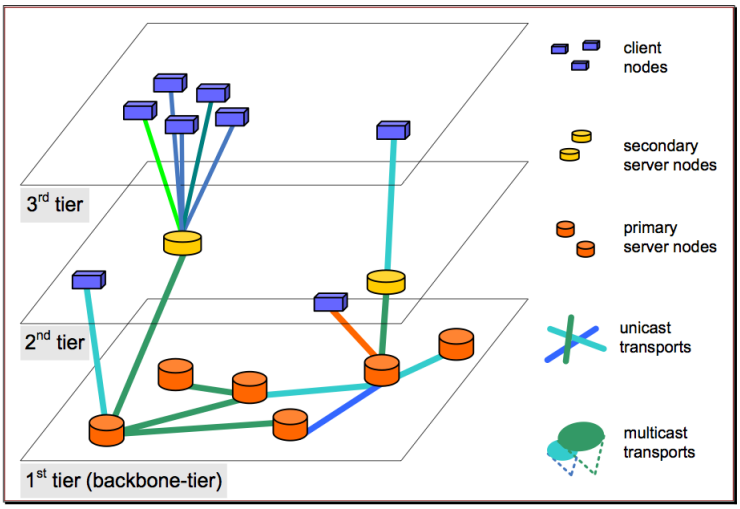


Figura 4.2: Arquitectura de três camadas de FEEDS - retirado de [16]

A primeira camada, denominada de *backbone tier*, é constituída unicamente por servidores primários, e o seu objectivo consiste em formar e gerir toda a rede *overlay*. Esta cama representa o ponto central da plataforma, representando o conjunto de entidades sementes às quais todos os demais nós constituintes da plataforma de devem ligar.

A segunda camada é composta primariamente por servidores secundários, podendo igualmente comportar nós clientes ainda que em número reduzido. O objectivo desta camada consiste em assegurar a capacidade de escala da rede, libertando a *backbone tier* da gestão directa dos nós clientes.

A terceira e última camada é unicamente composta por nós clientes, e assenta sobre as duas camadas anteriormente descritas para efeitos de comunicação, isto é, os nós clientes dependem das duas camadas anteriores para conseguirem fazer a descoberta e/ou comunicação com os demais nós clientes que constituem esta camada.

A filosofia principal da plataforma FEEDS consiste em suportar a coexistência de vários canais de eventos, caracterizados por qualidades de serviço particulares. Estas qualidades de serviço são definidas à custa de *templates* que consistem em módulos de abstracção que determinam um conjunto de regras de endereçamento. Assim, cada canal consiste numa instancia de um *template* que dita as qualidades de serviço asseguradas para esse mesmo canal, conseguidas a pedido. Por essa razão são denominados de canais de eventos activos.

Os *templates* podem ser encarados como o conjunto de regras que ditam a organização da rede *overlay*, uma vez que estes definem a forma como os eventos são encaminhados pela rede. Ou seja, a rede *overlay* pode ter diferentes configurações dependendo dos *templates* que são instanciados (na forma de canais).

Ao ser possível a existência de diferentes canais que instanciam diferentes *templates*, significa que podem coexistir diferentes configurações de rede simultaneamente, como ilustrado na figura 4.3.

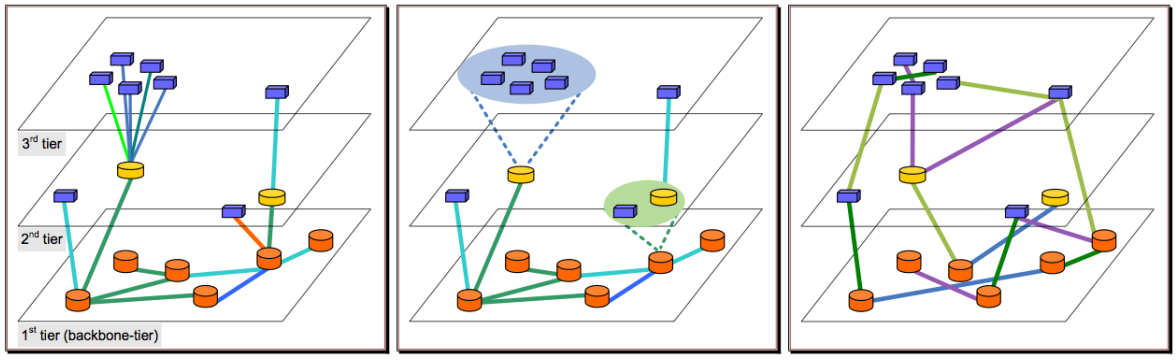


Figura 4.3: Exemplos de configurações possíveis de FEEDS - retirado de [16]

Como exemplos de diferentes configurações de rede possíveis, podemos considerar 3 *templates* que definem regras de endereçamento por forma a implementar diferentes modelos de comunicação, como *Peer-To-Peer*, Cliente/Servidor ou árvore de difusão. Todos os 3 modelos poderão coexistir num mesmo nó, bastando para isso que este instancie 3 canais referente a cada um dos *templates*.

A definição de novas funcionalidades em FEEDS é feita através da criação de serviços e de canais. Os serviços consistem na agregação de um conjunto de operações que, tirando partido da interacção entre diversos componentes da plataforma, produzem alterações no estado do nó, ou seja, definem um comportamento capaz de operar de forma autónoma.

4.2.2 Arquitectura MEEDS

Uma vez que MEEDS consiste numa extensão a FEEDS, esta assenta sobre os mesmos princípios funcionais, ou seja, opera sobre *templates* para definir regras de endereçamento de eventos e sobre serviços para definir novas funcionalidades.

Porém, para além de serviços e *templates*, MEEDS introduz três novos tipos de entidades:

4.2.2.1 Homebase

A *homebase* consiste num nó cliente FEEDS que serve de elo de ligação e ponto de acesso da entidade móvel ao substrato *publish/subscribe*. É uma entidade conhecida do

nó móvel a si associado, sendo que cada entidade móvel tem sempre uma *homebase* a correr numa entidade fixa numa relação de 1 para 1, ou seja, cada entidade móvel tem apenas uma *homebase* associada e vice-versa.

A relação entre a entidade móvel e a *homebase* é justificada por dois factores principais, pois por um lado, permite que as lacunas em termos de disponibilidade de recursos da entidade móvel sejam complementadas por uma entidade fixa, e por outro, permite um aumento de eficiência uma vez que a entidade fixa tem um acesso privilegiado em termos de comunicações com as restantes entidades da infra-estrutura.

Como implementação futura, está previsto o suporte de mecanismos que permitam assegurar a persistência dos dados em trânsito com destino ao nó móvel quando este se encontra em período de desconexão.

4.2.2.2 Proxy

Os *proxies*, tal como as *homebases*, consistem em nós clientes FEEDS, e funcionam como servidores de *roaming* das componentes móveis das aplicações implementadas sobre MEEDS. Ou seja, podem ser encarados como um conjunto de *gateways* distribuídos geograficamente, que oferecem suporte de comunicação entre as entidades móveis e a infra-estrutura fixa. Estes definem zonas de alcance usadas para determinar quais os nós móveis que devem servir em qualquer altura. Por sua vez, à medida que se vão deslocando as entidades móveis vão mudando o *proxy* a que se ligam, por forma a manter ligação com o proxy mais próximo. Na impossibilidade de se ligar a um *proxy*, o nó móvel recorre à sua *homebase* para conseguir comunicar com a infra-estrutura.

Ao servirem de intermediários pontuais, os *proxies* conseguem diminuir a latência nas comunicações entre os nós móveis e a infra-estrutura, uma vez que estes passam a comunicar com uma entidade mais próxima da sua localização, o que se reflecte num aumento da qualidade de comunicação fornecida às aplicações.

4.2.2.3 Entidade Móvel

A entidade móvel representa tanto um nó físico como uma entidade lógica, isto é, introduz os dispositivos móveis dos utilizadores como novos nós clientes da plataforma, definindo para isso entidades lógicas que actuam ao nível destes dispositivos.

Estas entidades lógicas são conhecidas como *tunnel*, e consistem numa abstracção que representa o meio através do qual as entidades móveis comunicam com a infra-estrutura fixa. Estas têm a responsabilidade de tornar processo de comunicação eficiente e transparente para a entidade móvel, garantindo que esta comunica com a entidade fixa que lhe permite uma menor latência de comunicação. Ou seja, o *tunnel* tem

uma participação activa no processo de associação das entidades móveis aos *proxies*, com o objectivo de melhorar a eficiência das comunicações.

A figura 4.4 ilustra a arquitectura de MEEDS.

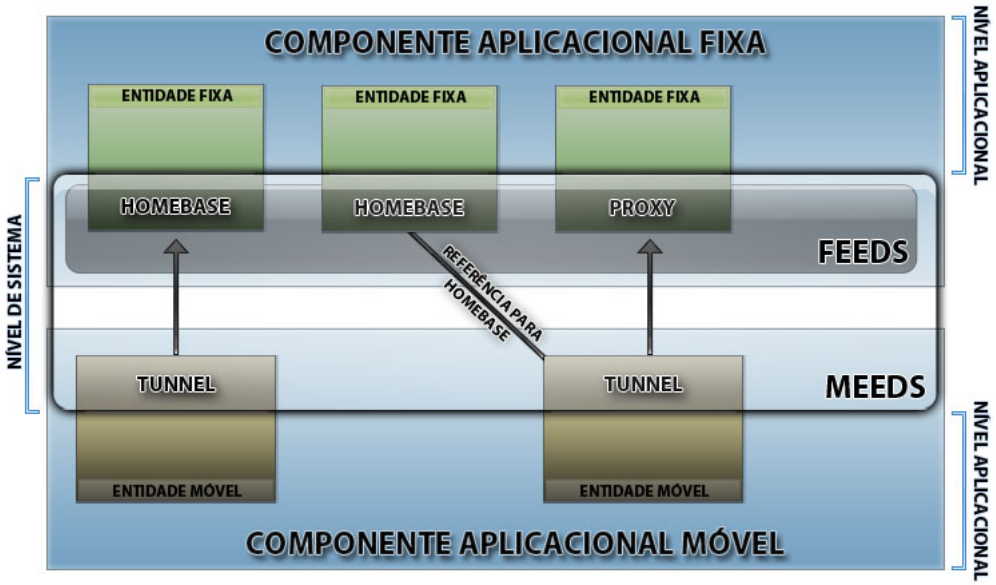


Figura 4.4: Arquitectura MEEDS

Tanto as *homebases* como os *proxies* representam conceitos e não a especificação de características de um nó físico, isto é, são representações de funcionalidades introduzidas por MEEDS para suporte de mobilidade. Estes conceitos são materializados na forma de serviços, pelo que um nó será considerado uma *homebase* se instanciar os serviços associados a este conceito, tal como acontece com os *proxies*. Assim, qualquer nó cliente FEEDS pode ser uma *homebase*, um *proxy*, ou ambos simultaneamente dependendo dos serviços que instancie.

No caso dos nós móveis, a situação difere uma vez que estes representam nós físicos que operam em dispositivos móveis dos utilizadores. Porém, estes assentam de igual forma sobre serviços e *templates* para suportar todas as funcionalidades, que definem tanto o processo de comunicação com a infra-estrutura, como as suas responsabilidades para com as aplicações.

Na figura 4.5, apresenta-se uma nova ilustração da arquitectura de MEEDS, com o intuito de elucidar melhor a relação existente entre FEEDS e MEEDS. Nesta, FEEDS é apresentada como sendo responsável pela infra-estrutura fixa, actuando ambos os *proxies* e as *homebases* como nós clientes FEEDS, que oferecem serviços de suporte à mobilidade das componentes móveis.

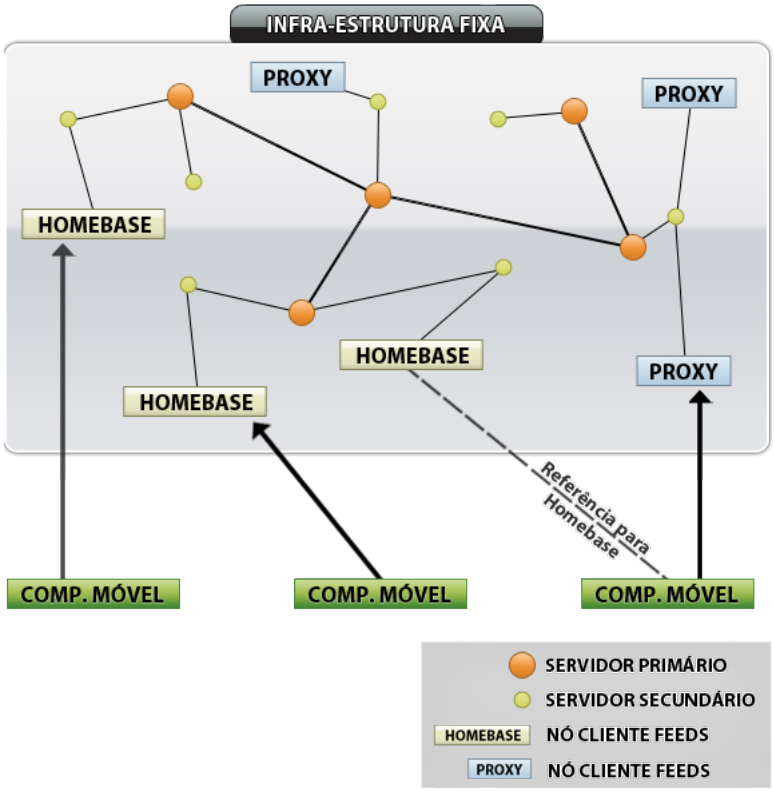


Figura 4.5: Arquitectura MEEDS

4.2.3 Modelo de Execução

MEEDS suporta dois modelos de execução transparentes para as aplicações, podendo estas operar em modo real ou em modo simulado. O modo simulado destina-se a fins de testes e avaliação de desempenho de aplicações, pois num ambiente de operação real este constitui usualmente um problema, uma vez que é bastante mais oneroso recriar de forma repetida um controlo sobre o comportamento dos indivíduos.

Para suportar a execução em modo simulado, MEEDS implementa um simulador que suporta a execução de múltiplos nós sobre uma máquina virtual Java.

Ao suportar ambos os modos de execução, MEEDS possibilita que qualquer aplicação consiga operar de igual forma tanto num contexto real como num contexto simu-

lado, sem necessidade de alterações à própria aplicação. A figura 4.6 ilustra o modelo de execução descrito.

Figura 4.6: Organização do modelo de execução

4.2.4 Modelo de Programação

A plataforma MEEDS está desenhada por forma a retirar o máximo possível a responsabilidade do processo de encaminhamento de eventos das aplicações. Para tal, esta suporta, como já foi referido previamente, o conceito de canal de eventos. Por esta razão, a API disponibilizada pela plataforma é significativamente simplificada, como ilustrado na figura 4.7.

ChannelDirectory ⟵ ***directory***()
Channel ⟵ ***clone***(template, name,...)
Channel ⟵ ***lookup***(name, ...)
Receipt ⟵ ***publish***(envelope, payload)
Receipt ⟵ ***feedback***(receipt, envelope,payload)
void ⟵ ***reRoute***(receipt)
void ⟵ ***subscribe***(criteria, handback, handler)
void ⟵ ***unsubscribe***(handback)
void ⟵ ***subscribeFeedback***(criteria,handback, handler)
void ⟵ ***unsubscribeFeedback***(handback)
⟶ ***notify***(receipt, envelope, payload)
⟶ ***notifyFeedback***(receipt, envelope, payload)

Figura 4.7: API disponibilizada por MEEDS

Para proporcionar o encaminhamento com filtragem baseada no conteúdo, MEEDS define os eventos como sendo constituídos por duas partes, nomeadamente *envelope* e *payload*. O *envelope* é de acesso público pelo substrato de comunicação, ao passo que o *payload* apenas pode ser acedido pela aplicação, e consequentemente não pode ser usado para fins de encaminhamento do evento. Assim, o processo de filtragem, que usa os filtros (*criterias*) definidos em tempo de subscrição dos canais, é feito sobre o conteúdo do *envelope* dos eventos.

Como é já conhecido a este ponto, MEEDS segue aquilo que se poderia chamar um modelo *publish/subscribe/feedback*, pelo que existem três operações primárias de interacção com a plataforma. As operações de *publish* e *subscribe* correspondem aquelas encontradas num modelo *publish/subcribe* normal, sobre as quais os nós subscrevem canais, aguardando pela publicação de eventos. A operação de *feedback* é adoptada em MEEDS como uma forma de encaminhar respostas a eventos concretos, de volta ao editor.

Na listagem listagem 4.1 encontram-se exemplificadas as principiais operações possíveis de efectuar sobre a plataforma. Nestas destacam-se a operação de instanciação de canais, por meio da instrução de *lookup*, as operações de *publish* e *subscribe* para publicar e escutar por eventos sobre um canal, e as operações de *feedback* e *subscribeFeedback* que permitem publicar e escutar respostas a eventos.

Ambos os métodos de *subscribe* e *subscribeFeedback* consistem em métodos anónimos, uma vez que instanciam objectos anónimos que permitem ao programador estipular a forma como, tanto os eventos recebidos como as respostas aos mesmos vão ser processadas.

Listing 4.1: Principais operações de MEEDS

```
//LOOKUP
Channel<HBEvent, Void, HBReplyEvent, Void> hbCh = Meeds.lookup("/hbChannel/requests");

//PUBLISH
HBEvent eventToHB = new HBEvent(...);
hbCh.publish(eventToHB, null);

//SUBSCRIBE
Receipt receitp;
hbCh.subscribe(new Subscriber<HBEvent, Void>() {
    @Override
    public void notify(Receipt r, HBEvent e, Payload<Void> p) {
        receitp = r;
        ...
    }
});
```

```
//FEEDBACK
HBReplyEvent replyToRequeirer = new HBReplyEvent(...);
hbCh.feedback(receitp, replyToRequeirer, null);

//SUBSCRIBE FEEDBACK
hbCh.subscribeFeedback(new FeedbackSubscriber<HBReplyEvent, Void>() {
    @Override
    public void notifyFeedback(Receipt r, HBReplyEvent e, Payload<Void> p) {
        ...
    }
});
```

4.3 Caso de Estudo: *Car Pooling*

O protótipo foi desenvolvido de forma direccionada para o *car pooling*, procurando contextualizar o problema em análise nos seus diferentes cenários. Assim, neste ponto são descritos os requisitos que uma aplicação deste tipo apresenta.

Numa aplicação de gestão de boleias a pedido, temos duas classes participantes com requisitos distintos. Temos os utilizadores que solicitam boleias, **Requisitantes**, e os utilizadores que concedem essas mesmas boleias, **Condutores**.

- **Requisitantes**

  Do ponto de vista do requisitante, os objectivos são concretos: encontrar uma boleia do ponto A para o ponto B, dentro de um período de tempo claramente determinado. Os requisitos prendem-se com a capacidade de obter *feedback* sobre a viabilidade dos seus pedidos em tempo útil. Este tempo útil deve sempre ser o mais prontamente possível, garantindo uma margem de tempo suficiente para planear alternativas caso seja necessário.

- **Condutores**

  Do ponto de vista do condutor, a necessidade principal centra-se na localização eficiente dos requisitantes, no sentido de minimizar a margem possível de ocorrência de desencontros.

  Tendo em conta a capacidade de deslocação dos condutores, é certo que a mobilidade dos mesmos não pode ser controlada, no entanto esta não ocorre de forma aleatória. Assim, faz sentido assumir que estes usualmente conhecem de antemão os trajectos que vão percorrer, ainda que de forma grosseira, podendo estes ser calculados a partir de um *software* de navegação instalado nos dispositivos móveis.

4.4 Soluções Implementadas

Nesta secção, descreve-se o processo de implementação das soluções idealizadas para o problema em análise. Estas consistem nas soluções apresentadas na secção 3.3.2, modeladas em função dos 3 cenários apresentados na secção 3.2. Na tabela tabela 4.1, apresenta-se a associação entre os referidos cenários e as soluções propostas.

Cenário 1	Solução 1.A - Centrada nos Entidades Fixas Solução 1.B - Centrada nas Eventos
Cenário 2	Solução 2 - Mobilidade Singular
Cenário 3	Solução 3 - Mobilidade Conjunta

Tabela 4.1: Associação entre cenários e soluções propostas

Com o objectivo de simplificar a leitura do documento, definem-se as entidades ***Requisitantes*** para representar os utilizadores a operar enquanto requisitantes de boleias, e ***Condutores*** para representar utilizadores a operar enquanto condutores.

Para efeitos de comunicação, definem-se três canais de eventos comuns a todas as abordagens. São estes: ***RequestsChannel*** para solicitar boleias, ***OffersChannel*** para oferta de boleias e ***DealsChannel*** utilizado para fazer o acordo final da boleia entre o requisitante e o condutor. Os eventos, as entidades que actuam como *publishers* e *subscribers* e os filtros de subscrição dos canais variam de solução para solução.

Uma vez que os utilizadores recorrem a dispositivos móveis para interagir com o sistema, procura-se uma gestão de recursos eficiente. Para tal, os utilizadores tiram partido das *homebases* a si associadas, delegando, sempre que faça sentido, tarefas que necessitam ver realizadas. Desta forma, o dispositivo móvel actua enquanto mecanismo de interacção para colheita e disponibilização de dados ao utilizador. A comunicação entre o nó móvel e a sua *homebase* é feita por meio de um canal, implementado para o efeito, designado ***HomebaseChannel***. Este canal simula a comunicação ponto-a-ponto, uma vez que envolve unicamente a comunicação entre uma componente móvel e a componente fixa a si associada (*homebase*).

Uma vez encontrados os potenciais parceiros para definir uma boleia, o nó requisitante inicia o processo de acordo de boleia, que se processa de igual modo em todas as abordagens. Por esta razão, este processo é descrito numa secção distinta (secção 4.4.5).

Nos anexos A.1, A.2, A.3, A.4 e A.5 apresentam-se as tabelas sumárias dos canais subscritos em cada uma das soluções posteriormente descritas.

4.4.1 Solução 1.A - Centrada nas Entidades Fixas

Esta solução consiste na implementação da abordagem baseada nas entidades fixas, referida na solução proposta 1 (ponto 3.3.2.1). Como foi referido na sua descrição, esta solução é referente ao cenário 1, no qual se ignora a mobilidade dos utilizadores para definir as áreas de interesse. No contexto do caso de estudo apresentado no ponto anterior, isto implica que tanto os pedidos como as ofertas de boleia recaem sobre pontos geográficos concretos, sendo independentes da mobilidade dos utilizadores que os definem.

A figura 4.8, ilustra o diagrama funcional da solução, onde são representados, quer os canais subscritos, quer o fluxo de mensagens definidos pela solução.

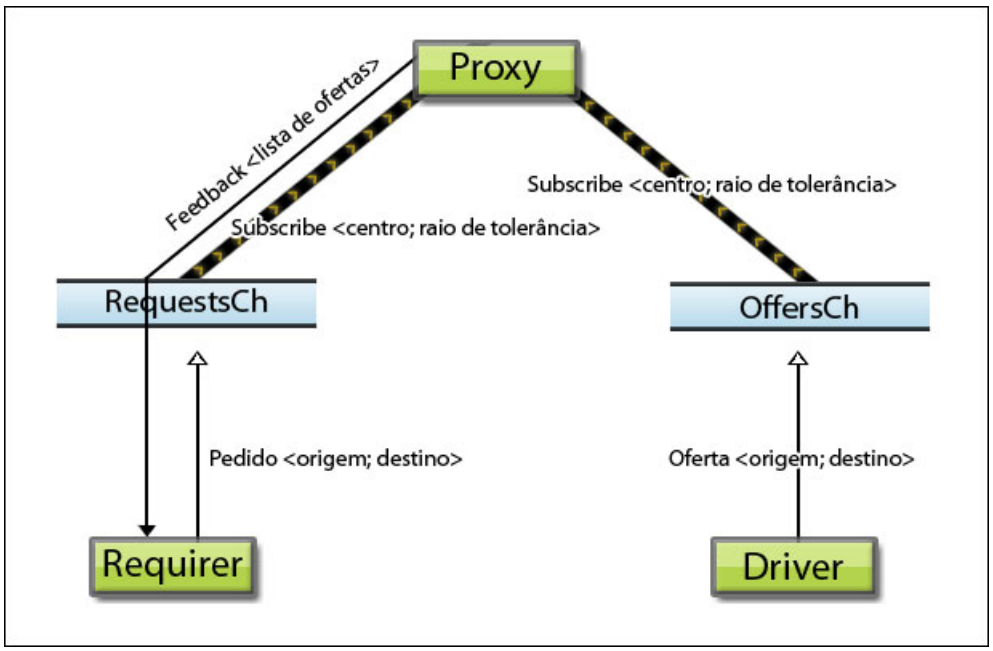


Figura 4.8: Diagrama funcional da solução 1.A - Aproximação baseada nas entidades fixas

Todos os ***proxies*** que pretendam colaborar na estipulação de boleias, subscrevem ambos os canais *RequestsChannel* e *OffersChannel* com um filtro correspondente a um círculo centrado na sua posição geográfica de raio r (parametrizado).

Para efectuarem pedidos de boleias, os requisitantes publicam no canal *RequestsChannel* eventos contendo os dados do utilizador, a sua localização geográfica, a posição de destino para onde se pretende deslocar, o tempo de duração do pedido e um valor representativo da distância máxima aos condutores. Este último é utilizado

para descartar todos os condutores que se encontrem demasiado distantes do requisitante. Caso não ocorra uma resposta de um *proxy* a garantir a persistência do pedido, desenrola-se um caso especial descrito mais à frente na presente secção.

Por sua vez, para efectuarem ofertas, os condutores publicam no canal *OffersChannel* eventos que contêm os dados do utilizador, a sua localização geográfica, a posição de destino para onde se vai deslocar e o tempo de duração da oferta.

Para efeitos de filtragem, em ambos os canais *RequestsChannel* e *OffersChannel* são usadas as posições de destino transportadas nos eventos, para determinar se o *proxy* vai aceitar ou rejeitar o evento. Desta forma, apenas os eventos cujas posições de destino se encontrem no interior do círculo de subscrição do *proxy* serão aceites pelo mesmo, o que implica que cada *proxy* fica responsável por uma zona geográfica.

Sempre que um *proxy* receba um novo evento através do *RequestsChannel*, isto é, sempre que receba uma nova solicitação de boleia, este processa a lista de ofertas que recebeu até então, para descartar todas aquelas cujas posições de origem do condutor estejam demasiado distantes da posição do requisitante. Este processo de triagem é feito com base no valor de distância máxima passado nos eventos de pedido de boleia. Uma vez determinadas as boleias prometedoras para o pedido recebido, o *proxy* responde por *feedback* ao requisitante, com a listagem destas ofertas, ou com uma mensagem vazia significando a inexistência de ofertas.

Sempre que surja uma nova oferta, o *proxy* verifica todos os pedidos para os quais a boleia é viável/negociável, e encaminha uma resposta por *feedback* para os respectivos requisitantes. Estes, após receberem a listagem de potenciais boleias, seleccionam uma e iniciam a fase de estabelecimento de acordo descrita na secção 4.4.5.

No caso em que existam dois *proxies* cujas áreas de filtragem se sobreponham numa zona geográfica sobre a qual é feito um pedido ou oferta de boleia, ambos asseguram a persistência desse evento. Significa isto que, no caso dos pedidos, a área que cobre eventuais ofertas passa a corresponder às zonas de filtragem de ambos os *proxies*, visto que ambos vão encaminhar as ofertas recebidas para o requisitante. No caso de se tratar de uma oferta, significa que esta tem uma maior probabilidade de ser encaminhada para um requisitante, pois ambos os *proxies* a usam para esse fim. A figura 4.9 ilustra a situação descrita.

O facto de existir duplicação na persistência de pedidos ou ofertas, não acarreta problemas funcionais uma vez que a escolha de ofertas recai sobre o requisitante, que se encarrega de filtrar as ofertas duplicadas.

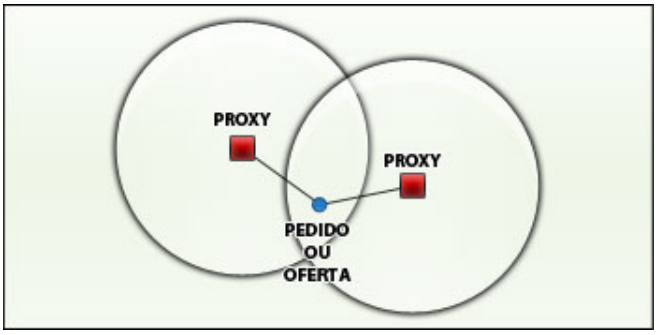


Figura 4.9: Solução 1.A - Intersecção de áreas de filtragem

CASO ESPECIAL

Pode no entanto ocorrer a situação em que um evento de pedido de boleia não coincide com nenhum círculo de filtragem de nenhum *proxy*. Para solucionar este problema, o requisitante delega a responsabilidade da persistência do pedido para a sua *homebase*. Para tal, este publica no canal *HomebaseChannel* um evento contendo todos os dados relativos ao pedido. A *homebase*, ao receber este evento, subscreve ambos os canais *RequestsChannel* e *OffersChannel* com um círculo de filtragem centrado na posição de destino do pedido e de raio r (parametrizado). Desta forma, a *homebase* fica responsável por quaisquer outros eventos que entretanto coincidam com esta zona geográfica, bem como pelos eventos que já haveriam sido publicados, uma vez que ambos os requisitantes e os condutores publicam os seus eventos periodicamente.

Assim, na ausência de um *proxy* que garanta a persistência de um pedido de um requisitante, a *homebase* deste assume esse papel comportando-se como se de um *proxy* localizado na posição de destino do pedido se tratasse. A figura 4.10 ilustra o processo de delegação de persistência de pedidos entre o requisitante e a sua *homebase*.

Quando um *proxy* ou uma *homebase* (a actuar enquanto *proxy*) detectam que uma oferta ou um pedido pelo qual estão responsáveis atinge o seu limite de duração, estes removem-no da lista. No caso de se tratar de uma *homebase*, quando se detecta que ambas as listas de ofertas e de pedidos ficam vazias, esta cancela a subscrição de ambos os canais *RequestsChannel* e *OffersChannel*, findando o seu comportamento enquanto *proxy* temporário.

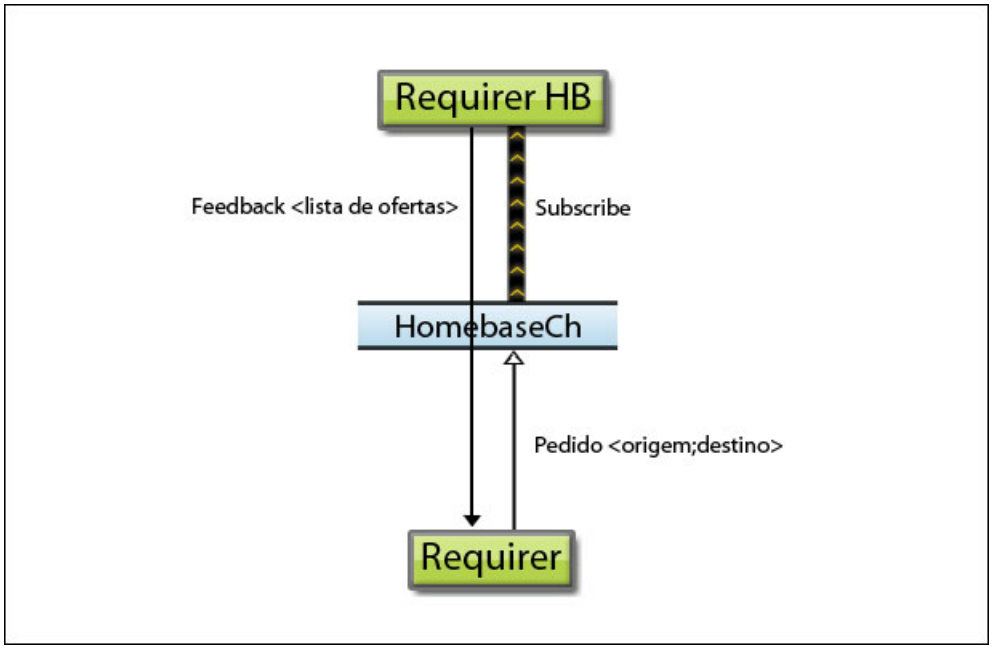


Figura 4.10: Delegação de tarefa para a *Homebase*

4.4.2 Solução 1.B - Centrada nos Eventos

Nesta solução descreve-se a implementação da segunda alternativa apresentada na solução proposta 1 (ponto 3.3.2.1), baseada nas mensagens. Nesta, e ao contrário da solução anterior, a complexidade do processo de responsabilização dos *proxies* pelos pedidos e ofertas de boleias é transportada para as mensagens. Significa isto que, em vez de serem os *proxies* a definir uma área de filtragem, que cubra a zona circundante da sua posição para determinar os eventos pelos quais se responsabilizam, passam a ser os eventos a transportar o raio de tolerância para o mesmo efeito. Este raio determina a distância ao *proxy* que se deve responsabilizar pelo evento. Assim, se um evento não receber resposta de um *proxy* a garantir a persistência do mesmo, este é novamente publicado com um raio de tolerância maior. Este processo é repetido até que um *proxy* responda ao evento ou até que se atinja um raio máximo.

A figura 4.11, ilustra o diagrama funcional da solução, onde são representados, quer os canais subscritos, quer o fluxo de mensagens definidos pela solução.

Todos os ***proxies*** que pretendam colaborar no estabelecimento de boleias, subscrevem ambos os canais *RequestsChannel* e *OffersChannel*, com um filtro baseado nas suas localizações geográficas.

Para efectuarem pedidos de boleias, os requisitantes publicam no canal *RequestsChannel* eventos onde é transportada informação relativa ao utilizador, a sua posição actual, a posição geográfica para onde o pedido é pretendido, o tempo de duração do pedido, o raio de tolerância para efeitos de filtragem e um valor representativo

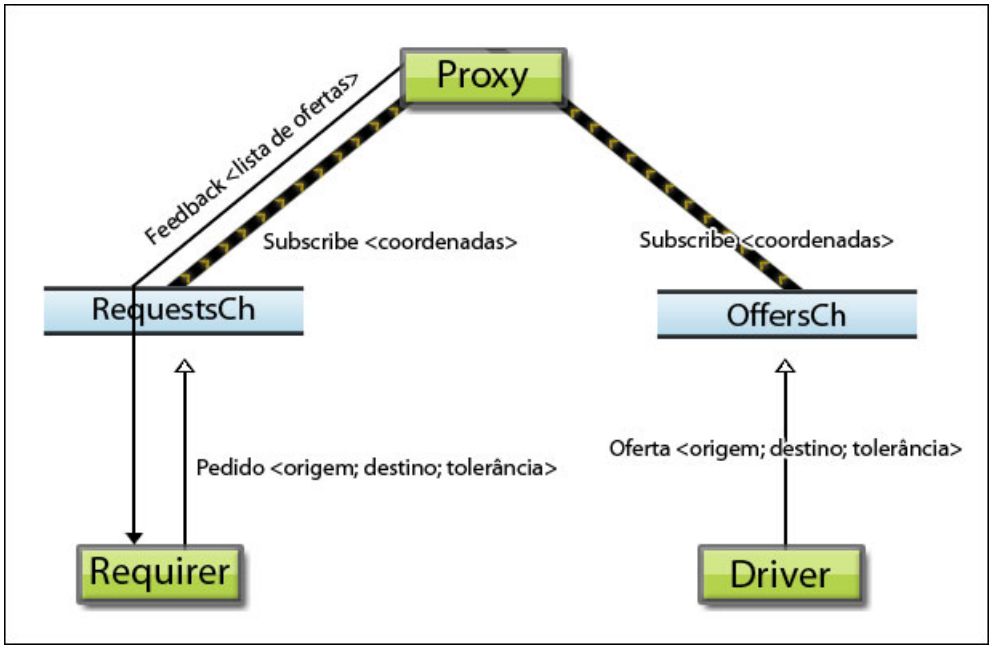


Figura 4.11: Diagrama funcional da solução 1.B - Aproximação baseada nas mensagens

da distância máxima aos condutores. Este último é utilizado para descartar todos os condutores que se encontrem demasiado distantes do requisitante.

Para publicarem ofertas, os condutores publicam no canal *OffersChannel* eventos idênticos aos publicados pelos requisitantes (porém contendo dados referentes ao condutor e à oferta do mesmo), excepto em relação ao raio de tolerância, uma vez que o condutor não necessita de estabelecer uma distância máxima aos requisitantes pois essa avaliação é deixada ao critério do utilizador que na fase de acordo.

Para efeitos de filtragem, em ambos os canais *RequestsChannel* e *OffersChannel* são definidos círculos de tolerância baseados na informação transportada nos eventos. Estes são centrados nas posições de destino, e têm um raio igual ao raio de tolerância transportado nos mesmos. Uma vez que os filtros de subscrição definidos pelos *proxies* se baseiam nas suas posições geográficas, então todos aqueles cujas posições de filtragem se encontrem no interior de um círculo de tolerância de um evento, deverão assegurar a persistência do mesmo.

A figura 3.9 apresentada na secção 3.3.2.1, ilustra o cenário descrito.

Sempre que um *proxy* receba um novo evento através do *RequestsChannel*, este processa a lista de ofertas que recebeu até então, para descartar todas aquelas cujas posições de origem do condutor estejam demasiado distantes da posição do requisitante. Este processo de triagem é feito com base no valor de distância máxima passado nos

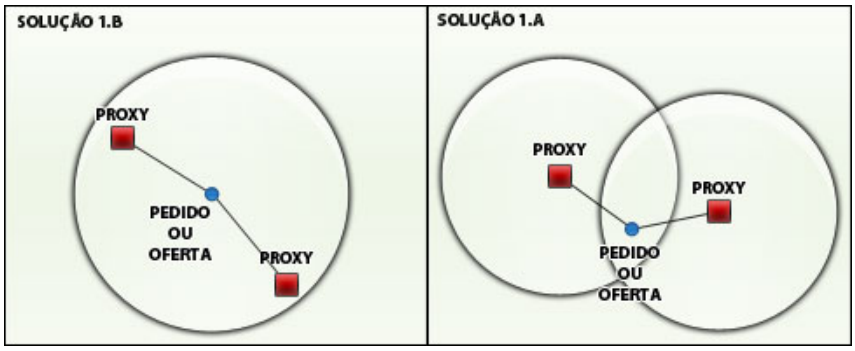
eventos de pedido de boleia. Uma vez determinadas as boleias prometedoras para o pedido recebido, o *proxy* responde por *feedback* ao requisitante, com a listagem destas ofertas, ou com uma mensagem vazia significando a inexistência de ofertas.

Sempre que surja uma nova oferta, o *proxy* verifica todos os pedidos para os quais a boleia é viável/negociável, e encaminha uma resposta por *feedback* para os respectivos requisitantes. Estes, após receberem a listagem de potenciais boleias, seleccionam uma e iniciam a fase de estabelecimento de acordo descrita na secção 4.4.5.

No caso em que existam dois *proxies* que se localizem no interior do círculo de filtragem de um mesmo evento, ambos asseguram a persistência deste. Assim, no caso de se tratar de um pedido, ambos os *proxies* respondem com a lista de ofertas compatíveis, sendo da responsabilidade do requisitante detectar eventuais ofertas duplicadas. Caso se trate de uma oferta, ambos os *proxies* armazenam os seus dados na lista de ofertas.

A figura 4.12a ilustra o círculo de tolerância correspondente a um evento. Este evento fica persistente em ambos os *proxies* cujas localizações se encontram no interior do círculo.

A figura 4.12b corresponde à figura 4.9 apresentada no ponto anterior. Comparando esta com a figura 4.12a, pode-se observar a semelhança na lógica que sustenta ambas as soluções 1 e 2, pois o processo de filtragem em ambas é idêntico, isto é, em ambos os casos este processo baseia-se na intersecção entre um ponto geográfico e um círculo que cobre uma área geográfica.



(a) Solução 1.B (b) Solução 1.A

Figura 4.12: Comparação de lógica funcional entre as soluções 1.A e 1.B

Pode no entanto ocorrer o cenário em que o círculo de tolerância transportado no evento não contém a posição geográfica de nenhum *proxy*. No entanto, uma vez que os dados referentes ao círculo são transportados pelo evento, a sua alteração implica apenas a publicação de um novo evento em que a única coisa que difere para o anterior é apenas o raio de tolerância, que é aumentado. Assim, se ao final de um período

de tempo pré-estipulado, o nó móvel que publicou o evento não receber qualquer notificação, este assume que não existe nenhum *proxy* localizado no interior do círculo de tolerância, e passa a publicar o mesmo evento mas com um valor de raio maior. Este processo repete-se até o nó móvel receber uma notificação de um *proxy*, ou até ser atingido um máximo de incremento para o raio de tolerância, sendo que neste caso o evento não fica persistente em nenhum *proxy*.

Este processo pode ser observado na figura 3.10 apresentada na secção 3.3.2.2, onde são ilustrados os raios de tolerância para cada evento.

4.4.3 Solução 2 - Mobilidade Singular

A solução 2 corresponde à implementação da solução proposta 2, descrita na secção 3.3.2.2. Esta recai sobre o cenário 2, no qual se contempla a mobilidade dos condutores para o processo de determinação de viabilidade de boleias. Significa isto que a definição de áreas de interesse, usadas para determinar os condutores e os requisitantes que estão em condições de estabelecer uma boleia, terá em consideração unicamente a mobilidade dos condutores. Desta forma, os pedidos de boleia mantêm-se, tal como nas duas soluções anteriores, independentes da mobilidade dos utilizadores que os definem.

Nestas circunstâncias, para que uma boleia seja possível, é necessário que a deslocação do condutor se dê de acordo com as necessidades do pedido, isto é, no sentido origem -> destino do mesmo, e que passe geograficamente perto destes pontos.

A figura 4.13, ilustra o diagrama funcional da solução, onde são representados, os canais subscritos e o fluxo de mensagens definidos pela solução.

Quando um requisitante pretende fazer uma solicitação de boleia, publica no canal *HomebaseChannel* um evento contendo informação relativa ao utilizador, a sua posição geográfica, a localização do destino para onde se pretende mover, o raio de tolerância e a duração do pedido. A *homebase*, ao receber um evento de pedido do requisitante a si associado, subscreve o canal *OffersChannel* usando como filtro os dois círculos, centrados nos pontos de origem e destino do pedido, e de raio igual ao raio de tolerância.

Por sua vez, para efectuarem ofertas, os condutores publicam periodicamente no canal *OffersChannel* um evento contendo os dados do utilizador, a sua localização geográfica e a lista dos pontos interpoladores da sua trajectória. Esta lista de pontos vai sendo cada vez menor conforme os eventos são publicados pelo condutor, uma vez que à medida que este se vai movendo, os troços por onde já passou deixam de fazer sentido para a trajectória.

Se a trajectória de um condutor intersectar ambos os círculos que servem de filtro

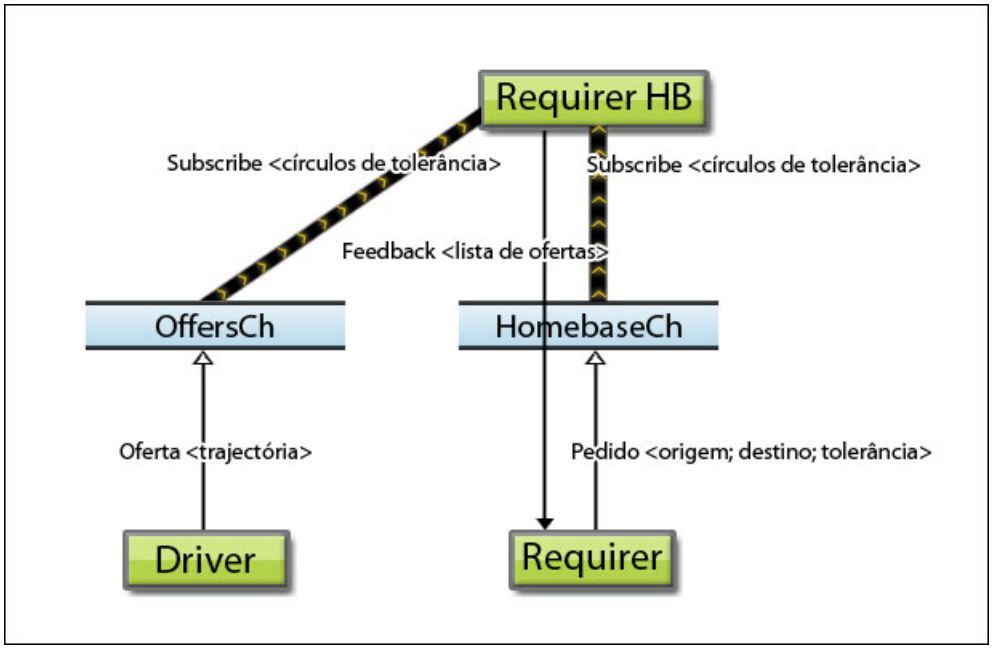


Figura 4.13: Diagrama funcional da solução 2

de subscrição de uma qualquer *homebase* de um requisitante, e se a deslocação se der no sentido origem -> destino do pedido, então o evento de oferta é aceite pela *homebase* em questão, e seguidamente encaminhada para o respectivo requisitante a si associado.

O requisitante, ao receber ofertas de boleias a partir da sua *homebase*, selecciona uma e inicia a fase de estabelecimento de acordo descrita na secção 4.4.5.

O facto de ser o próprio condutor a publicar periodicamente no canal *OffersChannel* os eventos correspondentes à actualização da sua mobilidade, pode sugerir uma gestão ineficiente de recursos. Isto porque se trata de um dispositivo móvel, que tem uma *homebase* associada, para a qual pode delegar esta tarefa. No entanto, tendo em conta que se assume que os condutores se deslocam a uma velocidade elevada, isto implicaria que estes teriam que publicar eventos no canal *HomebaseChannel* com uma regularidade igualmente elevada. Desta forma, esta opção tornar-se-ia mais pesada pois os número de eventos publicados seria duplicado, uma vez que a *homebase* teria apenas a função de repetidor de eventos.

4.4.4 Solução 3 - Mobilidade Conjunta

Nesta secção descreve-se a implementação da solução proposta 3, descrita na secção 3.3.2.3. Esta solução recai sobre o cenário 3, no qual se contempla a mobilidade de ambos os condutores e os requisitantes para o processo de determinação de viabilidade de boleias. Isto implica que a definição de áreas de interesse, usadas para determinar

utilizadores que estão em condições de estabelecer uma boleia, terá em consideração quer a mobilidade dos condutores quer a dos requisitantes. Porém, assume-se que apenas os condutores conhecem de antemão os percursos que vão seguir.

Tal como na solução anterior, para que uma boleia seja possível, é necessário que a deslocação do condutor se dê de acordo com as necessidades do pedido, isto é, no sentido origem -> destino do mesmo, e que passe geograficamente perto destes pontos.

A figura 4.14, ilustra o diagrama funcional da solução, onde são representados, quer os canais subscritos, quer o fluxo de mensagens definidos pela solução.

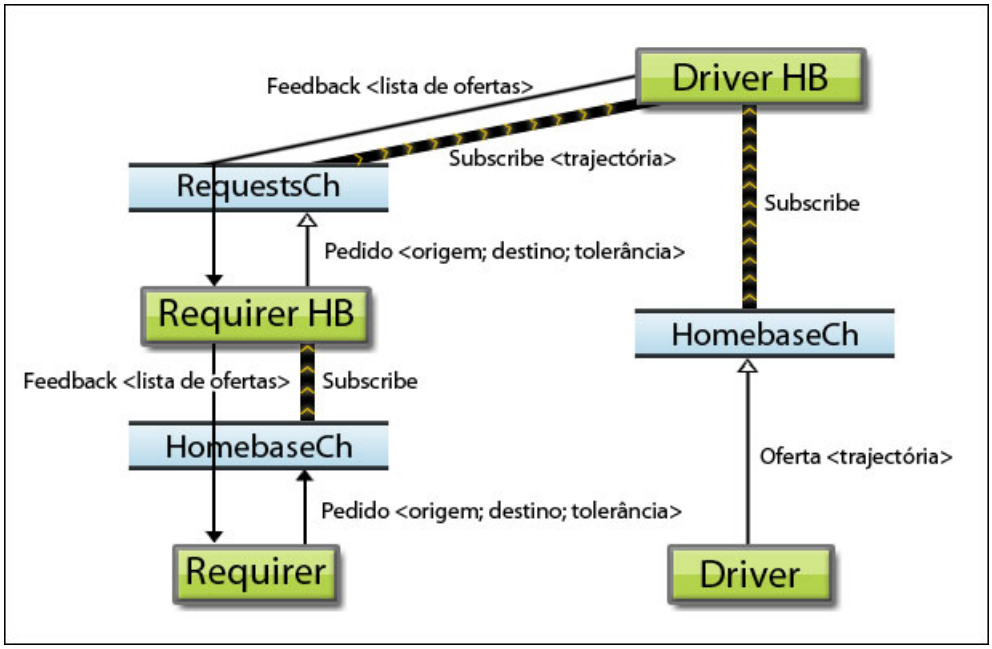


Figura 4.14: Ilustração de abordagem 4 - *Full Mobility Approach*

Quando um condutor pretende fornecer uma boleia, publica um evento no canal *HomebaseChannel* contendo os dados do utilizador, a sua localização geográfica e a lista dos pontos interpoladores da sua trajectória. A *homebase*, ao receber a oferta, subscreve o canal *RequestsChannel* com um filtro correspondente à lista de pontos interpoladores da trajectória. Após publicar o primeiro evento que transporta todos os dados da oferta, o condutor mantém a publicação de eventos de actualização, para que a sua *homebase* consiga saber o estado da sua mobilidade. Estes transportam unicamente a posição do condutor.

Por sua vez, quando um requisitante pretende solicitar uma boleia, remete o pedido para a sua *homebase* através de um evento publicado no canal *HomebaseChannel*. Neste evento são transportados os dados do utilizador, a sua localização geográfica, a locali-

zação geográfica do destino para onde se pretende deslocar, um raio de tolerância e o tempo de duração do pedido. Para manter a sua posição actualizada, o requisitante repete a publicação deste evento, ainda que com uma regularidade baixa, pois assume-se que este se desloca a uma velocidade reduzida.

Quando uma *homebase* associada a um requisitante recebe um evento através do canal *HomebaseChannel*, esta publica-o periodicamente no canal *RequestsChannel* até terminar o período de duração do pedido, ou até receber um novo evento relativo ao mesmo pedido, que actualiza a posição do requisitante, e que substitui o anterior.

Se a trajectória usada como filtro de subscrição de uma qualquer *homebase* associada a um condutor, intersectar ambos os círculos centrados na origem e no destino de um pedido, com raio igual ao raio de tolerância, e se o sentido da trajectória ocorrer no sentido origem -> destino do pedido, então a *homebase* do condutor recebe o pedido. Após receber o pedido, a *homebase* do condutor efectua uma pós-filtragem, para descartar todos os pedidos cujas posições de origem se localizem perto de um troço pelo qual o condutor já passou. Através deste processo, consegue-se contornar a necessidade de actualização frequente de filtros de subscrição, pois o filtro continua a abranger toda a trajectória do condutor.

Caso um pedido passe no processo de pós-filtragem, a *homebase* do condutor responde por *feedback* com os dados da trajectória. A *homebase* que publicou o pedido, após receber a oferta, encaminha-a para o respectivo requisitante, que se encarrega de escolher entre todas as ofertas recebidas e de iniciar a fase de estabelecimento de acordo descrita na secção 4.4.5.

No final da solução anterior descreve-se a justificação para o facto de o condutor publicar directamente no canal *OffersChannel*. Na presente solução, podemos observar que o requisitante toma um comportamento semelhante a este, no entanto, este delega a responsabilidade da publicação periódica para a sua *homebase*. Este facto é justificado pela velocidade de deslocação do requisitante, pois uma vez que este se desloca a uma velocidade reduzida, a frequência com que a sua posição necessita de ser actualizada relativamente ao seu pedido, é igualmente reduzida. Desta forma, durante o período de tempo em que o requisitante se encontra sem actualizar a sua posição, a sua *homebase* encarrega-se de publicar o pedido no canal *RequestsChannel*, baseando-se na última posição conhecida do requisitante. Com isto, pretende-se maximizar a probabilidade de encontrar ofertas viáveis para os pedidos, uma vez que a publicação frequente dos mesmos diminui a possibilidade de ocorrer o desencontro com uma oferta viável que entretanto deixe de o ser.

4.4.5 Negociação de Acordo

Ao iniciar a fase de acordo, é garantido que o requisitante já conhece todos os condutores em condições de oferecer uma boleia que cumprem os requisitos do seu pedido. No contexto do problema em análise na presente dissertação, significa que a entidade móvel já conhece os parceiros com quem deve comunicar. A fase de negociação de acordo consiste portanto na realização final da comunicação entre a entidade móvel e os seus parceiros.

Desta forma, após tomar conhecimento de todas as possibilidades de boleia, o requisitante deve fazer a eleição daquela que se aparenta mais viável, e comunicar ao respectivo condutor a sua intenção de negociação de boleia. Para tal, este publica no canal *DealsChannel* um evento que transporta o seus dados, a referência do evento de oferta a que corresponde o pedido de boleia e o ID do Condutor. Por sua vez, aquando do seu arranque inicial, o condutor subscreve o canal *DealsChannel* usando o seu ID como filtro de subscrição. Ao receber um pedido de acordo, o condutor toma a decisão e responde for *feedback* com um evento contendo a resposta ao pedido, e no caso positivo, os dados referentes ao ponto de encontro.

A figura 4.15, ilustra o diagrama funcional da fase de acordo, onde são representados, quer os canais subscritos, quer o fluxo de mensagens definidos pela solução.

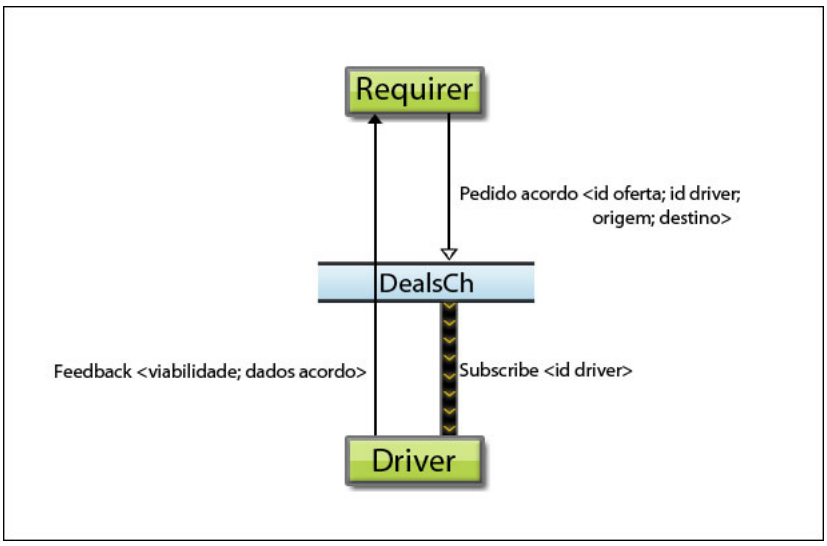


Figura 4.15: Diagrama funcional da negociação de acordo

5

Validação Experimental

Neste capítulo descreve-se a metodologia experimental que guiou o processo de validação das soluções desenvolvidas. Este processo assenta na execução sobre um ambiente simulado, pois apenas desta forma se consegue um total controlo sobre o ambiente de operação.

Cada solução é avaliada segundo um conjunto de métricas, que permitem determinar dados estatísticos sobre diferentes aspectos. Com base nestas métricas, pretende-se comprovar tanto a viabilidade das soluções para a resolução do problema do encontro de entidades, como avaliar a eficácia e eficiência das mesmas. Assim, as referidas métricas avaliam critérios como o número de boleias acordadas, o número médio de mensagens trocadas para o estabelecimento de boleias ou a carga suportada por cada uma das entidades fixas no suporte a este serviço. Este processo avaliativo é descrito em maior detalhe na secção 5.2 do presente capítulo.

5.1 Ambiente de Execução

O ambiente de simulação usado foi conseguido com base no simulador disponibilizado por MEEDS. Este permite a definição de diversas configurações de rede, possibilitando a parametrização do número de nós envolventes e a especificação do seu comportamento. Este comportamento é ditado pelo tipo de entidade que o nó representa, podendo esta ser relativa à *backbone* da infra-estrutura fixa herdada por FEEDS, ou qualquer uma das três componentes inseridas por MEEDS.

Uma vez que o protótipo desenvolvido é baseado no contexto do *car pooling*, definiram-se dois tipos de nós móveis referentes aos ***Condutores*** e aos ***Requisitantes***. Estes representam as duas classes de utilizadores que participam no processo de estabelecimento de boleias, e como tal, apresentam requisitos distintos relativamente à mobilidade. Os requisitantes são caracterizados por uma baixa velocidade de deslocação, pois representam indivíduos que se deslocam a pé, podendo deslocar-se livremente em qualquer sentido. Por sua vez, os condutores deslocam-se com uma velocidade média elevada, estando no entanto limitados à deslocação sobre caminhos concretos, representativos das faixas de rodagem.

5.1.1 Estrutura de Suporte

A simulação do comportamento dos condutores, implicou a adição do conceito de estrada ao simulador. Para tal, foi necessário mapear uma zona geográfica numa estrutura de pontos que pudesse ser inserida no contexto de simulação. Isto foi conseguido através da ferramenta **JOSM** [32], que consiste num editor de mapas desenvolvido em Java que opera sobre o **OpenStreetMap** [33]. Esta permite decompor áreas geográficas num conjunto de nós e caminhos representados num ficheiro XML, sendo que um nó é composto, de entre outros dados, pelas coordenadas geográficas relativas ao ponto que mapeia.

O ficheiro XML é posteriormente processado, de forma a que os nós e caminhos sejam usados para criar um grafo não orientado, em que a função de custo, usada na criação de arestas, é baseada nas distâncias entre os nós. No entanto, uma vez que a localização dos nós é dada em coordenadas geográficas, foi necessário implementar um sistema de conversão que permitisse transformar coordenadas geográficas em coordenadas cartesianas baseadas nas dimensões da área de simulação. Na figura 5.1 é apresentada uma captura do ambiente de simulação, que incorpora o grafo correspondente a uma zona da baixa de Lisboa, e que foi utilizado em todos os testes de validação.

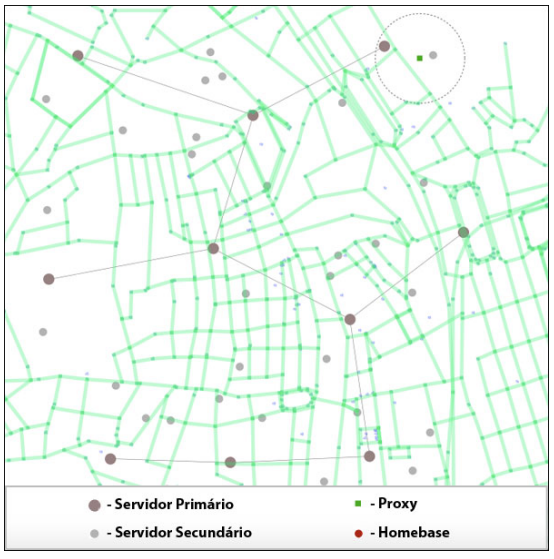
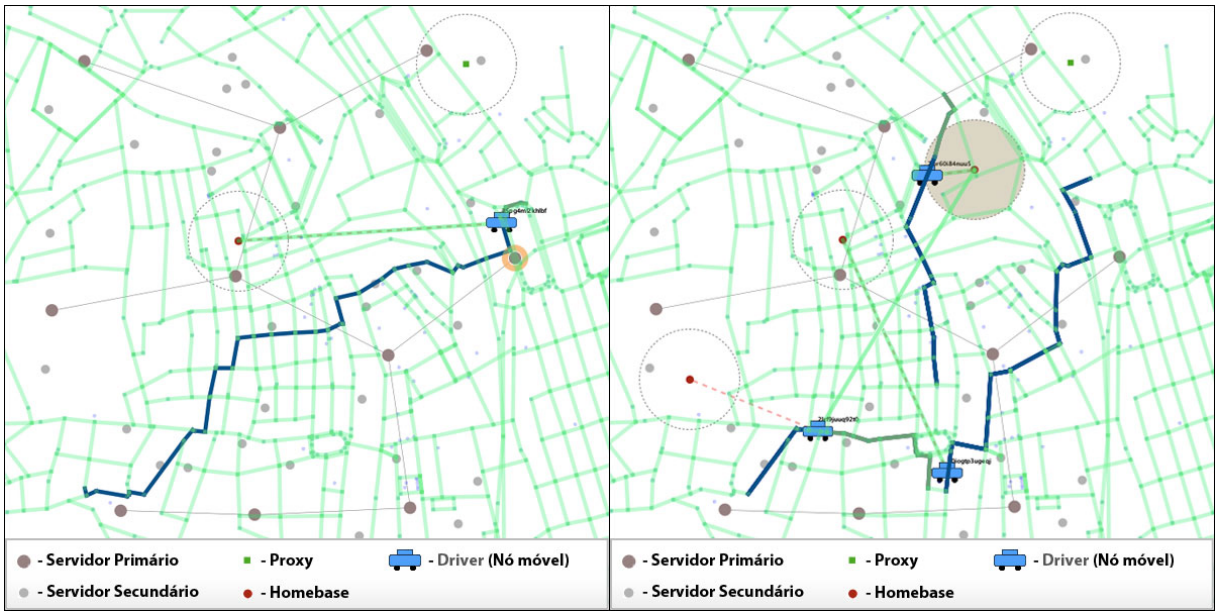


Figura 5.1: Mapeamento de zona geográfica num grafo não orientado

O grafo não orientado oferece a noção de estrada ao simulador. No entanto, para que seja possível a automatização das simulações, é necessário que os condutores se movam autonomamente sobre o ambiente de simulação. Para tal, foi criado um gerador de rotas aleatórias parametrizável, cujos parâmetros correspondem aos valores de máximo e mínimo de custo total das rotas geradas.

O funcionamento do gerador de rotas consiste na eleição aleatória de dois nós do grafo. Usando o algoritmo de *dijkstra* [14], encontra-se o caminho mais curto entre os dois nós. Se o custo total desse caminho estiver entre os limites parametrizados, a rota é válida. Caso contrário, elegem-se outros dois nós e repete-se o processo até encontrar uma rota válida. Na figura 5.2 são apresentados exemplos de rotas geradas.



(a) exemplo de rota com 1 condutor (b) exemplo de rota com 3 condutores

Figura 5.2: Exemplo de rotas geradas

5.1.2 Modelo de Mobilidade

O simulador suportado pela plataforma MEEDS implementa um modelo de mobilidade, que permite simular as deslocações dos nós móveis. Este modelo é definido com base numa política de atracção, segundo o qual os nós móveis são atraídos por entidades constituintes da plataforma, escolhidas aleatoriamente. Assim, os nós deslocam-se a uma velocidade constante até atingirem a localização da entidade para o qual estão a ser atraídos. Quando a distância entre o nó atraído e a entidade atractora é inferior a um valor pré-determinado, é eleito uma nova entidade atractora e o processo recomeça.

No entanto, para simular a mobilidade no contexto do *car pooling*, foi necessário alterar o mecanismo de atracção por forma a garantir a diferenciação de comportamento entre os **Requisitantes** e os **Condutores**. Ambos os tipos de intervenientes apresentam características de mobilidade únicas, caracterizadas da seguinte forma:

- **Requisitantes**

  Tendo em conta que estes nós representam indivíduos que se deslocam sem qualquer outro meio que não a locomoção, assume-se que estes se movem a uma velocidade de 6 Km/h, correspondente à velocidade média de locomoção humana.

  Quando efectuam um pedido de oferta, os requisitantes movem-se de forma aleatória pela área de simulação, pois assume-se que quando um indivíduo faz um pedido de boleia, este aguarda pela descoberta de uma potencial boleia em vez de iniciar uma viagem a pé até ao destino para onde se pretende deslocar. Desta forma, os requisitantes mantêm o mesmo comportamento implementado pelo simulador, no que respeita à eleição de nós atractores.

- **Condutores**

  Por representarem indivíduos que se deslocam por meio de veículos, assume-se que os condutores se deslocam a uma velocidade de 50 Km/h, correspondente ao limite máximo de velocidade em zonas urbanas.

  A mobilidade dos condutores é sempre efectuada sobre os troços do grafo, uma vez que estes representam estradas. Assim, em todas as soluções é garantido que a posição de destino de qualquer oferta será sempre referente a um nó do grafo.

  Relativamente aos caminhos percorridos, em todas as soluções é utilizado o algoritmo de *dijkstra* para encontrar o caminho mais curto entre a localização do condutor e o destino da sua oferta.

Assim, os condutores deslocam-se sempre ao longo das trajectórias geradas. Para alcançar este comportamento, definiram-se como atractores dos condutores todos os nós do grafo constituintes das trajectórias que estes seguem, o que implicou a organização destes numa lista de forma ordenada consoante o sentido da trajectória. Desta forma, sempre que o condutor atinge um nó atractor, é eleito o próximo nó da trajectória como atractor, e assim sucessivamente até atingir a posição de destino da oferta. Quando terminada uma rota, é gerada uma nova rota e o processo é novamente iniciado.

5.1.3 Simulação de Abordagens

Tendo em conta que se pretende avaliar cada uma das soluções com recurso a simulação, é necessário que estas consigam operar de forma autónoma. Para tal, foram criados geradores de pedidos e de ofertas de boleias para cada umas das soluções implementadas. Estes geradores funcionam tendo em conta a premissa imposta que, cada requisitante ou cada condutor só podem ter respectivamente um pedido ou uma oferta activas em qualquer instante. Para conseguir isto, foram definidas rotinas que, periodicamente verificam se existe algum pedido ou alguma oferta activas, e caso não exista procede à sua criação.

Para definir as referidas rotinas, foram utilizadas Tarefas (***Tasks***) suportadas pelo simulador. Estas são assíncronas, e consistem numa abstracção que permite substituir o papel dos *Threads*.

5.2 Métricas de Avaliação

Nesta secção são descritas as métricas utilizadas na avaliação das soluções implementadas. O processo de recolha de dados estatísticos foi implementado com base num *singleton*, no qual foram disponibilizados métodos através dos quais as entidades conseguiam participar na recolha de dados. É importante referir que os dados recolhidos não foram utilizados em situação alguma enquanto argumentos no processamento de qualquer das soluções, assegurando desta forma a operação isolada de cada uma das entidades instanciadas.

As medidas usadas nesta avaliação são as seguintes:

- **Taxas de boleias estabelecidas**

  A taxa de boleias estabelecidas consiste no principal critério de avaliação da eficácia das soluções propostas, pois esta representa directamente se a solução é capaz ou não de oferecer o serviço de descoberta de potenciais parceiros que devem comunicar a cada instante. Esta taxa é representada matematicamente pela expressão:

$$\text{Taxa de boleias estabelecidas} = \frac{\text{Número de boleias estabelecidas}}{\text{Total de pedidos em condições de estabelecer uma boleia}}$$

  Para efectuar esta medição, cada requisitante que efectue um novo pedido instancia uma nova entrada de pedido pendente no processador estatístico. Quando um requisitante recebe uma resposta positiva a um pedido de acordo de boleia, este usa o identificador do pedido para fechar o pedido pendente como boleia estabelecida. Se ao fim da duração do pedido este se mantiver pendente, este é fechado como pedido falhado, sendo gerado um novo pedido.

  Para medir o total de pedidos em condições de estabelecer uma boleia, sempre que é gerado um novo pedido ou uma nova oferta de boleia, este é comparado respectivamente com todas as ofertas ou todos os pedidos existentes. Desta forma, efectua-se um processamento paralelo que permite determinar de entre todo o universo de pedidos de boleias, quais aquelas que deveriam resultar no estabelecimento de boleia. Este processamento é efectuado no processador estatístico, e não interfere como o normal funcionamento das soluções.

  É importante referir que o valor obtido para o cálculo da taxa de bolais estabelecidas não é exacto, pois existem 2 situações que podem gerar falsos positivos. Estas são as seguintes:

  – Pode ocorrer a situação em que a janela temporal entre o instante em que um pedido ou uma oferta são gerados e o instante em que finda a duração de, respectivamente, uma oferta ou pedido compatíveis é demasiado curta para que seja estabelecida uma boleia. Porém, do ponto de vista do processador estatístico, todos os pedidos nestas condições serão considerados como possíveis de estabelecer boleia uma vez que este apenas avalia a compatibilidade directa, sem considerar a questão do tempo.

  - Quando é efectuada a comparação entre um novo pedido ou oferta gerada e respectivamente todas as ofertas ou pedidos existentes, todas as ofertas que são compatíveis com um determinado pedido resultam neste ser considerado como um pedido em condições de estabelecer uma boleia. Assim, pode acontecer o caso em que 2 pedidos são compatíveis com uma mesma oferta, e como tal são ambos considerados como pedidos possíveis quando apenas um deles pode ser satisfeito.

Apesar de poder ser afectada por falsos positivos, a taxa de boleias estabelecidas pode igualmente ser afectada por uma situação possível, pois uma vez que os requisitantes efectuam tentativas de acordo de boleia de entre todas as ofertas compatíveis que recebem, pode ocorrer a situação em que este tenta estabelecer acordo com condutores cujas ofertas já estão associadas a outros pedidos concorrentes, resultando na rejeição do mesmo. Assim, pode-se dar o caso em que após diversas tentativas rejeitadas de estabelecimento de acordo, ou o pedido ou as ofertas que ainda não estão em boleia findam a sua duração, pelo que a boleia não é estabelecida

Desta forma, o valor calculado para a taxa de boleias estabelecidas deve ser encarado como um minorante, pelo que na realidade, o valor exacto correspondente será sempre igual ou maior aquele obtido. Tratando-se este de um problema de optimização, e tendo em conta o tempo disponível para elaboração da dissertação, optámos por não calcular o valor exacto.

- **Número médio de Mensagens trocadas**

  A medição do número de mensagens consumidas no estabelecimento de boleias, permite inferir a complexidade que este processo impõe à rede sobreposta que sustenta a plataforma MEEDS. Se este implicar um elevado número de mensagens trocadas, tal poder-se-á reflectir tanto num consumo excessivo de recursos da rede, como na limitação da longevidade de autonomia dos dispositivos móveis.

  Para calcular a média de mensagens trocadas são considerados todos os pedidos que terminaram em boleias sucedidas, sendo que, para cada um deles, são contabilizadas todas as mensagens consumidas durante este processo. Tal consiste no somatório de todas as mensagens publicadas nos canais de eventos instanciados em cada uma das soluções.

  O facto de as soluções serem implementadas sobre MEEDS, permite que estas se abstraiam do tratamento de problemas relativos às comunicações, tais como

percas ou duplicação de mensagens. Isto porque, como já foi referido na secção 4.2, estas qualidades de serviço são asseguradas pelos canais de eventos activos.

- **Distribuição de Carga nas entidades fixas**

  A carga suportada por uma entidade fixa consiste na quantidade de eventos pelos esta se responsabiliza. Para efectuar este cálculo, cada entidade fixa instancia um contador que é incrementado cada vez que esta se responsabiliza por um novo pedido ou uma nova oferta de boleia. Nas soluções 1 e 2, o processo é simples, bastando incrementar o contador por cada novo evento recebido que ainda não tenha sido contabilizado. Na solução 2, o contador é incrementado de cada vez que a *homebase* do requisitante subscreve o canal de ofertas, significando que recebeu um novo pedido, e é incrementado novamente cada vez que recebe uma oferta por este canal. Finalmente, na solução 3, o contador é incrementado cada vez que a *homebase* do condutor subscreve o canal de pedidos, significando que recebeu uma nova oferta do seu condutor, e novamente cada vez que recebe um novo pedido de boleia através do canal de pedidos. No caso da *homebase* do requisitante, o contador é incrementado de cada vez que esta recebe um novo pedido de boleia por parte deste.

  A medição da média de eventos pelos quais cada entidade fixa é responsável, permite inferir a distribuição de carga pela rede sobreposta que sustenta a plataforma. A partir destes dados, é possível determinar se as soluções implementadas conseguem um bom aproveitamento dos recursos de que dispõem, ou, se pelo contrário, centralizam a complexidade da operação num conjunto minoritário de entidades fixas.

  Existe uma relação estreita entre o número médio de mensagens consumidas por cada estabelecimento de boleia, e o número de pedidos nos quais uma entidade fixa participa, pois a análise conjunta destas permite determinar aproximadamente a quantidade de trabalho executado pelas entidades fixas.

5.3 Resultados

Nesta secção são apresentados os resultados obtidos referentes às métricas apresentadas anteriormente, com base na simulação das soluções implementadas. Para tornar o processo de comparação entre as prestações de cada uma das soluções o mais justo possível, usaram-se os mesmos parâmetros para todos os aspectos comuns a todas as soluções sob a mesma base, Assim:

- São sempre instanciados 20 nós móveis, sendo que destes 10 operam enquanto requisitantes e os restantes 10 enquanto condutores;
- Cada simulação corresponde ao equivalente a um período de 12 horas de execução em ambiente real;
- O raio dos círculos de tolerância utilizados no processo de filtragem de eventos é de aproximadamente 90 metros. Cremos que este valor corresponde a um raio razoável, atendendo tanto às dimensões da área mapeada como à quantidade de nós e ligações que compõem o grafo derivado desta;

Tanto a solução 1.A como a solução 1.B são simuladas com base em 3 configurações distintas, diferindo estas relativamente ao número de *proxies* instanciados. O número de *proxies* a executar no contexto destas soluções, tem um impacto directo sobre a eficiência das mesmas, uma vez que estes participam no processo de estabelecimento de boleias. Desta forma, pretende-se avaliar o impacto que esta variação apresenta sobre a eficiência e sobre a eficácia de ambas as soluções.

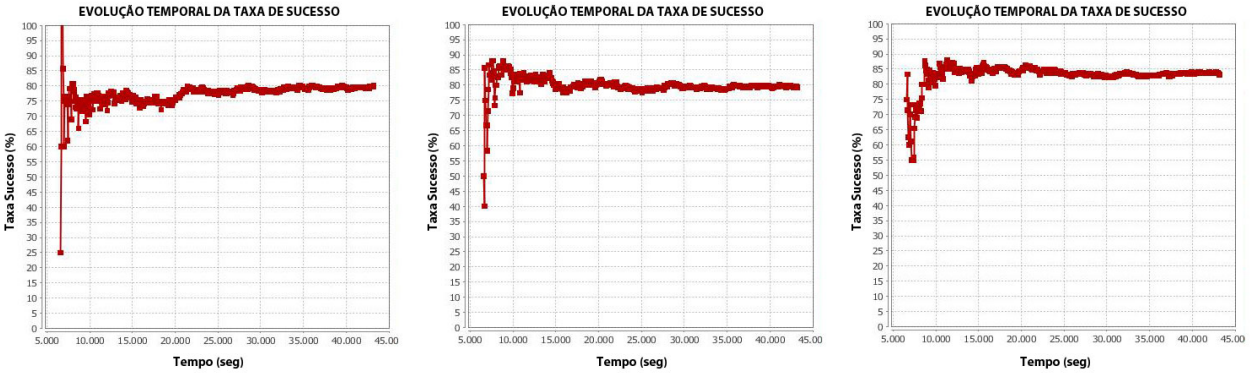
Na tabela 5.1 são descritas as configurações referidas anteriormente.

	Número de *Proxies*
Conf5	5
Conf20	20
Conf50	50

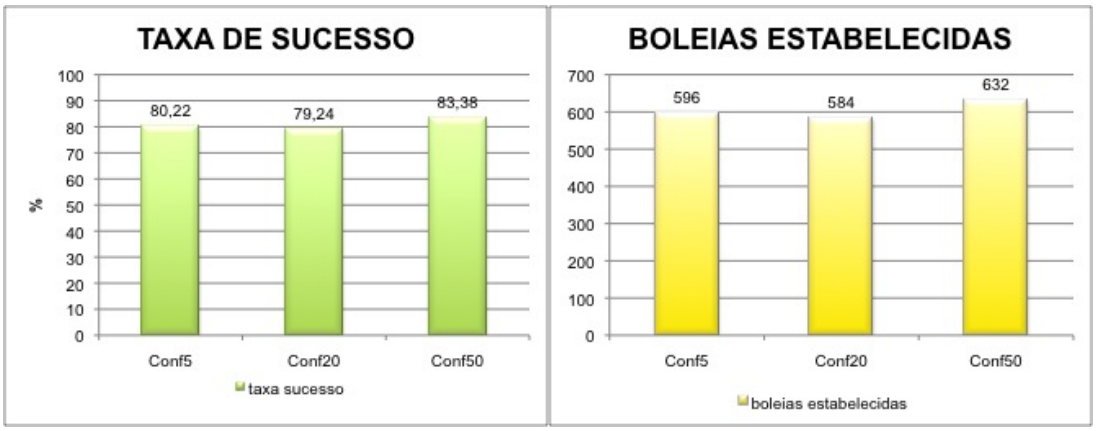
Tabela 5.1: Descrição das configurações usadas na avaliação das soluções 1 e 2

5.3.1 Resultados para Solução 1.A

TAXA DE BOLEIAS ESTABELECIDAS



(a) Progressão da taxa de sucesso - Conf 1 (b) Progressão da taxa de sucesso - Conf 2 (c) Progressão da taxa de sucesso - Conf 3



(d) Taxa Final de Simulação (e) Número de boleias estabelecidas

Figura 5.3: Taxa de boleias estabelecidas para cada configuração

Analisando as taxas de sucesso obtidas para cada uma das configurações, ilustradas na figura 5.3d, pode-se observar que a variação do número de *proxies* tem um impacto mínimo ou mesmo nulo sobre o desempenho da solução. Este facto era esperado, uma vez que na ausência de um *proxy* que se responsabilize por um pedido de boleia, o requisitante encaminha o pedido para a sua *homebase* que desempenha esse papel. Este facto é reforçado tanto pelos gráficos de progressão temporal das taxas de sucesso ilustradas nas figuras 5.3a, 5.3b e 5.3c, como pelo gráfico representante do número de boleias estabelecidas apresentado na figura 5.3e.

Desta forma, a *homebase* comporta-se como um *proxy* temporário, de forma totalmente transparente para todos os nós móveis que entretanto definam pedidos ou ofertas de boleia para essa zona geográfica. Assim, a solução consegue circunscrever o

problema da inexistência de *proxies*, garantindo que os pedidos de boleia são sempre atendidos por uma entidade fixa.

NÚMERO DE MENSAGENS POR BOLEIA ESTABELECIDA

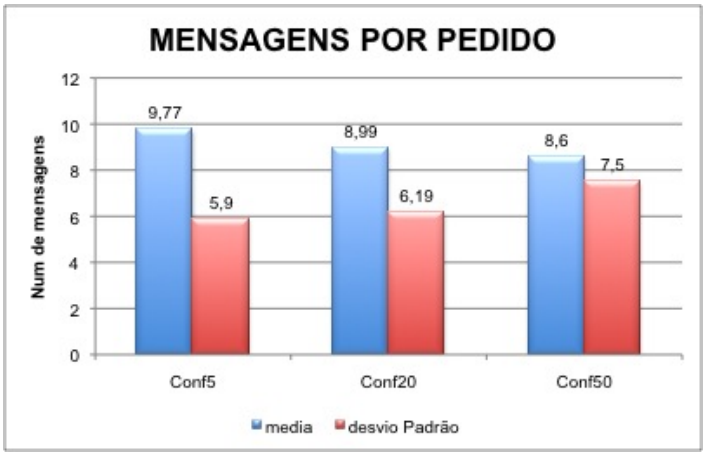


Figura 5.4: Mensagens Por Pedido

Observando o gráfico apresentado na figura 5.4, constata-se que com o aumento do número de *proxies*, ocorre uma diminuição no número médio de mensagens consumidas por cada boleia estabelecida. Tal comportamento é justificado pelo facto de, ao existirem mais *proxies*, existir igualmente uma maior área geográfica coberta pelos mesmos, o que se reflecte numa diminuição da necessidade de os requisitantes delegarem a responsabilidade de persistência dos seus pedidos para as suas *homebases*. Porém, se analisarmos o desvio padrão apresentado igualmente na figura 5.4, observa-se que com o aumento de *proxies*, este aumenta também. Isto é justificado pelo facto de a distribuição geográfica tanto dos *proxies* como das posições de destino dos pedidos e ofertas de boleia, não ocorrer de forma totalmente uniforme. Assim, podem existir zonas geográficas com maior concentração de *proxies*, e outras com uma maior concentração de *eventos*, podendo estas serem distintas. Tal implica que em determinadas zonas vai haver uma maior necessidade de delegar tarefas para as *homebases* do que noutras.

DISTRIBUIÇÃO DE CARGA POR ENTIDADES FIXAS

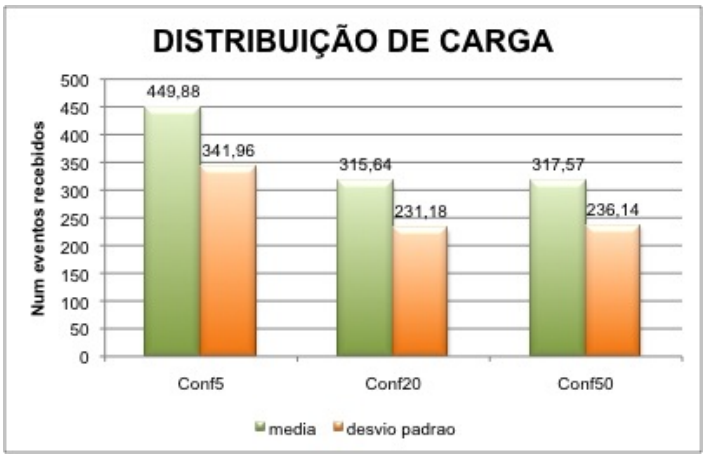


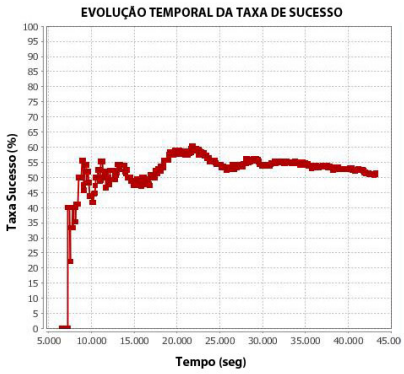
Figura 5.5: Distribuição de Carga por Entidades Fixas

Tal como acontece com o número médio de mensagens consumidas por boleia estabelecida, também a distribuição de carga diminui com o aumento do número de *proxies*, pois passa a existir um maior número de entidades fixas capazes de intervir no processo de estabelecimento de boleias. No entanto, comparando as configurações 2 e 3, observa-se que a distribuição de carga é semelhante, o que sugere que existe um valor óptimo para o número de *proxies*, que maximiza a distribuição de carga entre as entidades fixas. A justificação para tal facto prende-se com a sobreposição de áreas de tolerância, resultante do aumento do número de *proxies*. Nesta situação, ambos os *proxies* vão aceitar o evento, o que significa que ambos aumentam o número de eventos suportados.

5.3.2 Resultados para Solução 1.B

TAXA DE BOLEIAS ESTABELECIDAS

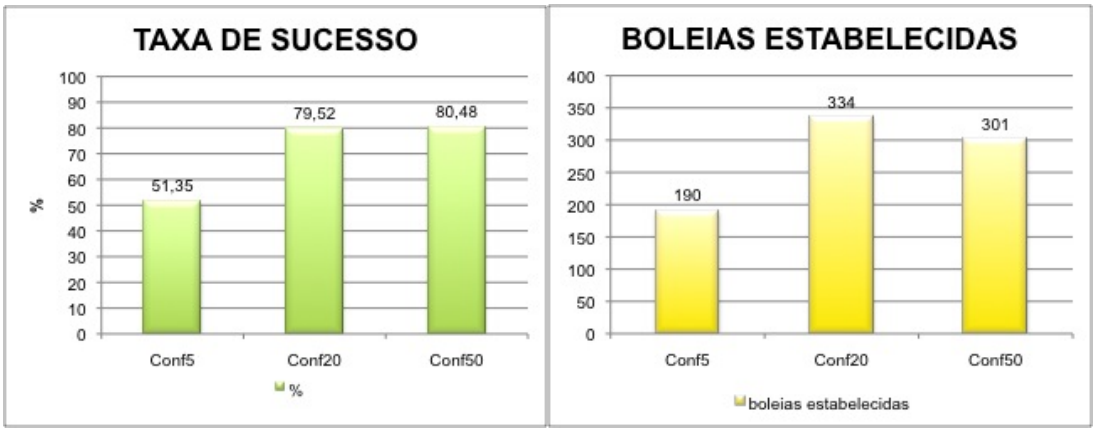
Relembrando a solução 1.B, esta é caracterizada pelo aumento progressivo do raio de tolerância publicado nos eventos de pedido, até que este inclua um *proxy* ou até ser atingido um raio máximo. Se analisarmos as taxas de sucesso obtidas para cada uma das configurações, ilustradas nas figuras 5.6a, 5.6b, 5.6c, 5.6d com base na lógica da solução, facilmente se compreende a razão para os resultados obtidos na conf1 serem substancialmente inferiores aqueles obtidos para as restantes configurações. Tal deve-se ao facto de ocorrer frequentemente a situação em que os nós móveis esgotam



(a) Progressão da taxa de sucesso - Conf 1

(b) Progressão da taxa de sucesso - Conf 2

(c) Progressão da taxa de sucesso - Conf 3



(d) Taxa Final de Simulação

(e) Número de boleias estabelecidas

Figura 5.6: Taxa de boleias estabelecidas para cada configuração

o incremento do raio de tolerância dos seus eventos, e ainda assim não intersectam com nenhum *proxy*, dado que estes existem num número reduzido. Desta forma, existem grandes zonas geográficas que ficam por cobrir, sendo que qualquer evento que seja feito sobre a mesma, não será atendido por nenhuma entidade fixa, o que se reflecte num menor número de boleias estabelecidas, tal como ilustrado no gráfico apresentado na figura 5.6e.

Comparando os resultados obtidos para as configurações conf20 e conf50, podemos constatar que, tal como acontece na solução 1.A, um aumento do número de *proxies* não se reflecte necessariamente num aumento da eficácia. Assim, podemos inferir que para cada região geográfica existirá um valor óptimo do número de *proxies* que maximiza a eficácia da solução.

NÚMERO DE MENSAGENS POR BOLEIA ESTABELECIDA

Analisando no gráfico apresentado na figura 5.7, é notório o facto de que o aumento do número de *proxies* implica a diminuição do número médio de mensagens necessárias para o estabelecimento de uma boleia. Isto apesar de a taxa de boleias estabelecidas não evoluir com base na mesma relação, como se observou no ponto anterior. Tal é explicado pelo facto de que com o aumento do número de *proxies*, aumenta a probabilidade de um pedido de boleia ficar associado a mais que 1 *proxy*, o que implica uma maior probabilidade de ocorrerem mais ofertas viáveis para esse pedido, diminuindo desta forma a probabilidade de ocorrência de pedidos concorrentes para uma mesma oferta. No entanto, observando o desvio padrão, podemos constatar que esta situação não ocorre de forma generalizada, pois este não diminui com a mesma proporção que a média, sendo este comportamento justificado pela distribuição não uniforme tanto dos *proxies* como das posições de destino dos eventos publicados.

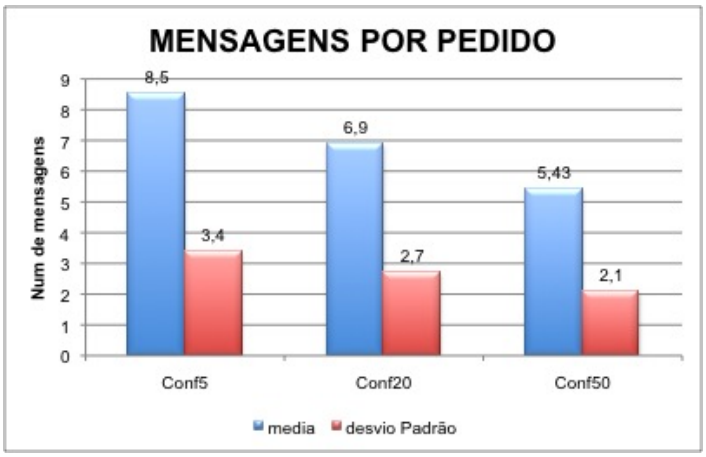


Figura 5.7: Mensagens Por Pedido

DISTRIBUIÇÃO DE CARGA POR ENTIDADES FIXAS

Tal como acontece com o número médio de mensagens consumidas no estabelecimento de boleias, também a distribuição média de carga pelas entidades fixas (no caso da presente solução estas são representadas unicamente por *proxies*) diminui com o aumento do número de *proxies* instanciados. Este facto é observado na figura 5.8, onde podemos constatar igualmente que, apesar de esta distribuição não ser perfeita, dados os valores de desvio padrão obtidos, esta é uniforme em todas as configurações, isto é, os valores de desvio padrão decrescem em proporção com a média de carga suportada pelas entidades fixas.

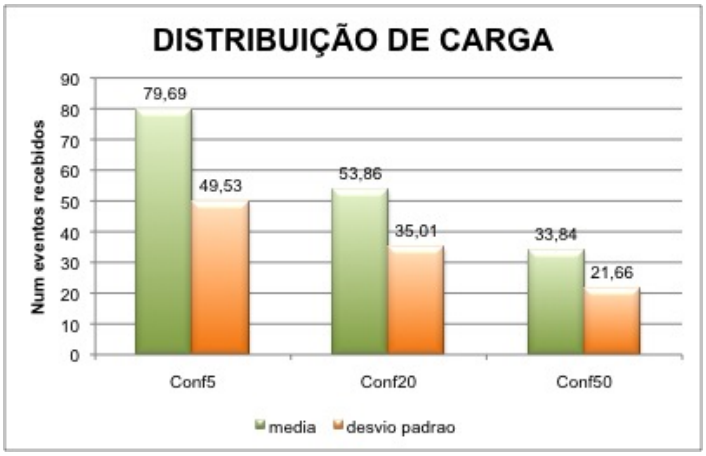


Figura 5.8: Distribuição de Carga por Entidades Fixas

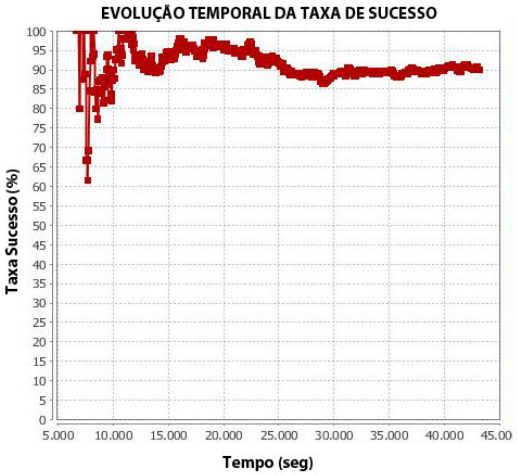
5.3.3 Resultados para Solução 2

Tendo em conta que a solução envolve unicamente as *homebases* dos requisitantes, e relembrando que cada entidade móvel instanciada implica a existência de uma e uma só *homebase*, não existe a necessidade de avaliar a presente solução com base em diferentes configurações.

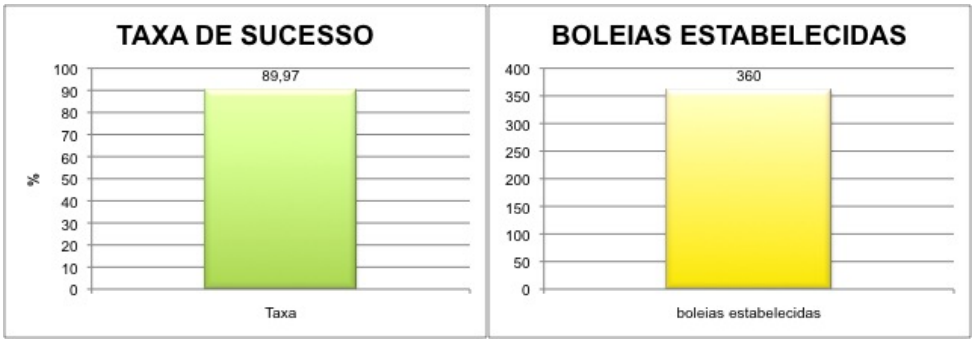
TAXA DE BOLEIAS ESTABELECIDAS

À imagem do que acontece na solução 1.A, também na presente solução os requisitantes delegam os seus pedidos para as respectivas *homebases* para que estas se encarreguem de os atender. Porém, contrariamente à solução 1.A, na presente solução este processo ocorre para todos os pedidos. Assim, se interpretarmos os resultados apresentados nos gráficos das figuras 5.9a e 5.9b com base na lógica apresentada, observamos que os resultados obtidos correspondem ao esperado, pois temos a garantia que cada pedido é sempre atendido por uma entidade fixa.

O facto de a taxa de sucesso neste cenário ser superior à observada para a solução 1.A é justificada pelo facto de ser necessária a verificação de um maior número de parâmetros, para que um pedido e uma oferta sejam considerados compatíveis. Isto implica que, comparativamente à solução 1.A, existe uma menor probabilidade de se encontrarem várias ofertas compatíveis para cada pedido. Tal significa que a probabilidade de ocorrência de falsos positivos gerados por pedidos concorrentes para uma mesma boleia é menor, assim como é menor a probabilidade de um requisitante esgotar a janela temporal disponível para estabelecer uma boleia com uma oferta compatível,



(a) Progressão da taxa de sucesso



(b) Taxa Final de Simulação (c) Número de boleias estabelecidas

Figura 5.9: Taxa de boleias estabelecidas para cada configuração

por gastar esse tempo a tentar estabelecer acordo com ofertas que entretanto já se encontram associadas a outros pedidos.

NÚMERO DE MENSAGENS POR BOLEIA ESTABELECIDA

Ao analisar os resultados apresentados no gráfico da figura 5.10, observa-se que a média de mensagens consumidas no estabelecimento de boleias não toma valores demasiado altos, ainda que, tendo em conta o desvio padrão, esta não apresente grande dispersão. Significa isto que terão existido boleias que consumiram um maior número de mensagens, contrariando outras que terão consumido um número de mensagens mínimo para a solução.

Se tivermos em conta que uma oferta e um pedido só são viáveis se: A trajectória da oferta intersectar ambos os círculos de tolerância circundantes das posições de origem e destino do pedido, se a trajectória se der no sentido origem->destino do pedido e se

o condutor não tiver ainda passado pelo troço que passa perto da origem do pedido, facilmente constatamos que a probabilidade de ocorrerem pedidos concorrentes para uma mesma oferta é bastante reduzida. Tal facto permite-nos inferir que o volume acrescido de mensagens consumidas no estabelecimento de algumas boleias terá sido fruto da actualização frequente da posição geográfica do condutor e não por tentativas sucessivas de estabelecimento de acordo entre requisitantes e condutores resultantes de pedidos concorrentes sobre ofertas.

Figura 5.10: Mensagens Por Pedido

DISTRIBUIÇÃO DE CARGA POR ENTIDADES FIXAS

Observando o desvio padrão apresentado na figura 5.11, é notória a baixa dispersão da distribuição de carga pelas entidades fixas (*homebases* dos requisitantes). Este comportamento é explicado pela facto de não existir uma distribuição uniforme na geração quer das posições de origem e destino dos pedidos, como das trajectórias seguidas pelos condutores. Significa isto que a ocorrência de pedidos e ofertas compatíveis ocorre de forma aleatória, e, tendo em conta que serão as *homebases* dos requisitantes a ficar responsáveis pelos seus pedidos, tal explica a baixa dispersão observada.

Figura 5.11: Distribuição de Carga por Entidades Fixas

5.3.4 Resultados para Solução 3

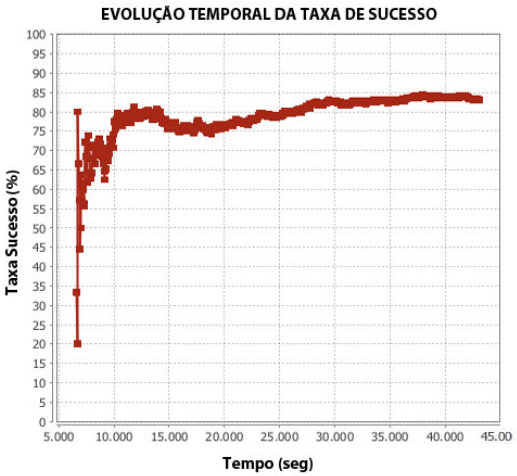
Tendo em conta que a presente solução envolve unicamente as *homebases* dos condutores, e relembrando que cada entidade móvel instanciada implica a existência de uma e uma só *homebase*, não existe a necessidade de avaliar a presente solução com base em diferentes configurações.

TAXA DE BOLEIAS ESTABELECIDAS

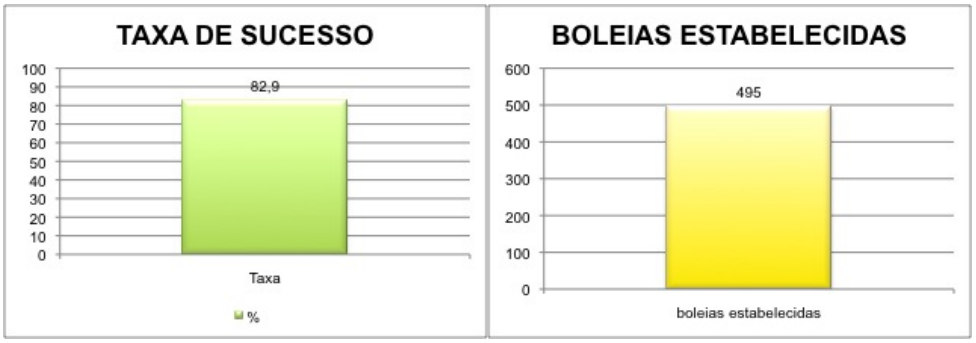
Relembrando a lógica que sustenta a solução 3, temos que sempre que um condutor inicia uma nova viagem, este comunica à sua *homebase* a rota que vai seguir, sendo que esta subscreve o canal de pedidos com um filtro baseado na trajectória do condutor.

Comparando a presente solução com a solução anterior, observamos que em ambas existe a garantia de atendimento para os eventos, ainda que em circunstâncias diferentes, na solução anterior são as *homebases* dos requisitantes que se responsabilizam pelos seus pedidos, ao passo que na presente solução são as *homebases* dos condutores que se responsabilizam pelas suas ofertas. Assim, também nesta solução são esperados valores elevados de taxa de sucesso de boleias estabelecidas, tal como comprovado pelos gráficos apresentados nas figuras 5.12a e 5.12b.

Porém, se compararmos os valores obtidos em ambas as soluções, observamos que este é mais baixo na presente solução que na solução anterior. Tal é explicado pela mobilidade do requisitante, pois quando a posição de origem dos pedidos se move, aumenta a probabilidade de ocorrerem pedidos concorrentes sobre a mesma oferta, o que implica um aumento do número de falsos positivos. Porém, esta mobilidade sugere igualmente um aumento do número de boleias estabelecidas, pois um pedido que inicialmente não era compatível com nenhuma oferta pode vir a sê-lo, caso a origem



(a) Progressão da taxa de sucesso



(b) Taxa Final de Simulação (c) Número de boleias estabelecidas

Figura 5.12: Taxa de boleias estabelecidas para cada configuração

do pedido se desloque, e passe a intersectar a trajectória de uma oferta que reuna as restantes condições necessárias para que seja considerada compatível com o pedido. Este facto é comprovado pela comparação dos gráficos apresentados nas figuras 5.9c e 5.12c.

NÚMERO DE MENSAGENS POR BOLEIA ESTABELECIDA

Dado que na presente solução é necessário actualizar frequentemente quer as posições do condutor quer do requisitante, será de esperar um consumo médio de mensagens no estabelecimento de boleias superior aquele obtido na solução anterior, pois nesta apenas o condutor tinha necessidade de actualizar frequentemente a sua posição. Tal facto é comprovado pelos resultados obtidos, apresentados na figura 5.13, que, como se pode depreender do baixo desvio padrão, consiste numa situação generalizada para a maioria das boleias estabelecidas.

Figura 5.13: Mensagens Por Pedido

DISTRIBUIÇÃO DE CARGA POR ENTIDADES FIXAS

So compararmos os resultados apresentados na figura 5.14 como os obtidos na solução anterior, observamos que tal como nesta, a dispersão da carga pelas entidades fixas (*homebases* dos condutores) é baixa, sendo este comportamento originado pelas mesmas razões descritas na solução anterior.

Figura 5.14: Distribuição de Carga por Entidades Fixas

5.3.5 Análise Crítica

Uma vez analisados os resultados obtidos para as simulações de cada uma das soluções desenvolvidas, apresenta-se na presente secção uma análise crítica sobre as prestações das mesmas.

As soluções 1.A e 1.B são aquelas que de todas melhor se podem comparar, uma vez que ambas são direccionadas sobre o mesmo cenário. Estas constituem, como foi já referido nas secções 4.4.1 e 4.4.2, numa inversão lógica uma da outra, isto é, ao passo que na solução 1.A são as entidades fixas que usam os círculos de tolerância enquanto filtros de subscrição, na solução 1.B essa informação é transportada para os eventos.

Comparando os resultados obtidos em cada uma destas, podemos concluir que apesar de apresentar uma taxa de boleias sucedidas mais consistente, e independente do número de *proxies* existentes, esta constitui uma opção mais cara e complexa. Isto porque esta implica não apenas um maior volume de mensagens consumidas por boleia estabelecida, como também uma frequente subscrição/dessubscrição de canais de eventos, o que como já foi referido diversas vezes no decorrer do presente documento, trata-se de uma operação dispendiosa e a evitar. Desta forma, faz sentido considerar a solução 1.B para cenários em que se sabe ser garantida a existência de um número mínimo de *proxies*, bem como a sua dispersão geográfica pela área em questão, uma vez que neste cenário consegue-se obter uma taxa de boleias estabelecidas semelhante àquela obtida caso se recorra à solução 1.A, porém com um consumo de recursos mais baixo.

No entanto, ao compararmos os resultados obtidos para o número de boleias estabelecidas nas soluções 1.A e 1.B, constatamos que este número é mais baixo no segundo caso. Tal é justificado por uma questão de configurações dos ambientes de teste, pois um a vez que na solução 1.B poderá ser necessário o aumento do raio de tolerância, optámos por aumentar a duração dos pedidos, o que, relembrando que cada requisitante só pode ter um pedido activo em qualquer instante, reflecte-se num menor número de pedidos efectuados, e consequentemente de boleias estabelecidas.

6

Conclusões

Neste capítulo são apresentadas as considerações finais sobre o trabalho realizado e os seus respectivos resultados, as contribuições proporcionadas e o possível trabalho futuro que possa surgir ou constituir um melhoria a esta dissertação.

6.1 Considerações Finais

Esta dissertação relata o objectivo de desenvolver um conjunto de soluções distribuídas para a problemática do encontro entre entidades móveis em ambientes distribuídos.

A partir da análise de um conjunto de trabalhos que se inserem no mesmo contexto de operação e que abordam, ainda que indirectamente, a mesma problemática, surgiu a motivação que seria possível desenvolver um conjunto de novas soluções direccionadas sobre aspectos que, a nosso ver, estão ainda pouco trabalhados. Estes aspectos prendem-se com a centralização da complexidade das soluções nas infra-estruturas fixas, pois acreditamos que, com base na crescente evolução dos dispositivos móveis, é já possível ponderar um cenário em que estes desempenham um papel mais activo. Desta forma, conseguem-se desenvolver soluções capazes de alcançar uma gestão de recursos mais eficiente ao relegar responsabilidades para os dispositivos móveis.

Porém, atribuir uma maior autonomia estes dispositivos implica que estes consigam encontrar os parceiros com quem devem comunicar a qualquer instante, com base em critérios temporais e geográficos. Assim, à complexidade que é relegada para os dispositivos móveis, acresce a necessidade de assegurar tais garantias o que pode demonstrar-se demasiado complexo de atingir em dispositivos que ainda apresentam grandes limitações em termos de disponibilidade de recursos.

Cremos que a solução para este problema consiste em desenvolver um ambiente simbiótico entre a infra-estrutura fixa e os dispositivos móveis de forma a que estes sejam suportados nas operações mais complexas. Desta forma, os dispositivos móveis podem concentrar-se no desempenho das suas tarefas, deixando a cargo da infra-estrutura fixa todo o papel de suporte e de intermediação no encontro dos parceiros com quem devem comunicar.

Em MEEDS vislumbrámos uma plataforma capaz de oferecer os requisitos necessários à criação de tal ambiente. Mais concretamente, propriedades de auto-organização, auto-regeneração, QoS ajustáveis às aplicações por intermédio da definição de *templates* e ainda um modelo de programação reactivo, com um baixo conjunto de primitivas, facilitando a curva de aprendizagem no que diz respeito ao uso do suporte.

Assim, partimos para o desenvolvimento de um conjunto de soluções assentes sobre os requisitos apresentados, sendo estas, no âmbito da presente dissertação, contextualizadas sobre a forma de um serviço de *car pooling*, com o objectivo de aproximar o problema em abstracto de um problema com contornos reais.

6.2 Contribuições

Atendendo ao trabalho realizado, e fazendo um balanço em termos de objectivos, cremos que estes foram alcançados, tendo resultado deste projecto 3 contributos importantes:

- As soluções desenvolvidas, que, com base nos resultados obtidos, comprovaram-se como opções válidas para a resolução do problema do encontro de entidade móveis em ambientes pervasivos.
- Um protótipo funcional que, pelas características implementadas, constitui uma ferramenta útil, tanto para a validação de aplicações *participatory sensing* desenvolvidas sobre MEEDS, como para projectos futuros de investigação que assentem sobre a mesma temática.

- Um conjunto de ferramentas, resultantes do processo de implementação do protótipo funcional, que servem de complemento ao simulador de MEEDS, e que introduzem ambos os conceitos de estrada e de mobilidade escalar.

6.3 Trabalho Futuro

Embora os objectivos propostos tenham sido, a nosso ver, alcançados nesta dissertação, temos consciência que alguns aspectos podem ser melhorados. Nesta secção enumeram-se aqueles que consideramos serem pontos de interesse para o desenvolvimento futuro sobre o trabalho já realizado.

- Desenvolver um conjunto de mecanismos que permitam um maior controlo sobre o sincronismo temporal na resolução do encontro entre parceiros;
- Refinamento do processo de validação das soluções, incluindo uma experimentação num ambiente real e outros contextos aplicacionais onde as soluções propostas se enquadrem;
- Integração das soluções enquanto módulos de MEEDS, por forma a disponibilizar uma interface genérica para o desenvolvimento de aplicações *participatory sensing*;

Anexos

Canal	***Publishers***	***Subscribers***	**Eventos**
Requests Channel	Requisitante	**Proxy**; Filtro: Círculo centrado na pos. geográfica do proxy de raio r **Homebase do Requisitante**; Filtro: Círculo centrado na pos. geográfica da *homebase* de raio r	***Publish*** - dados de utilizador - localização requisitante - destino do pedido - duração do pedido - dist. máx. aos condutores ***Feedback*** - lista ofertas
Offers Channel	Condutor	**Proxy**; Filtro: Círculo centrado na pos. geográfica do proxy de raio r **Homebase do Requisitante**; Filtro: Círculo centrado na pos. geográfica da *homebase* de raio r	***Publish*** - dados de utilizador - localização condutor - destino da oferta - duração da oferta ***Feedback*** - /
Homebase Channel	Requisitante	**Homebase do Requisitante**	***Publish*** - dados de utilizador - localização requisitante - destino do pedido - duração do pedido - dist. máx. aos condutores ***Feedback*** - lista ofertas

Tabela A.1: Tabela de sumária de canais subscritos na solução 1.A

Canal	***Publishers***	***Subscribers***	**Eventos**
Requests Channel	Requisitante	**Proxy**; Filtro: Posição geográfica do proxy	***Publish*** - dados de utilizador - localização requisitante - destino do pedido - raio tolerância - duração do pedido - dist. máx. aos condutores ***Feedback*** - lista ofertas
Offers Channel	Condutor	**Proxy**; Filtro: Posição geográfica do proxy	***Publish*** - dados de utilizador - localização condutor - destino da oferta - raio tolerância - duração da oferta ***Feedback*** - /

Tabela A.2: Tabela de sumária de canais subscritos na solução 1.B

Canal	***Publishers***	***Subscribers***	**Eventos**
Offers Channel	Condutor	**Homebase do Requisitante**; Filtro: Dois Círculos centrados na loc. geográfica do Requisitante e outro na pos. geográfica do destino do pedido, ambas de raio igual ao raio de tolerância	***Publish*** - dados de utilizador - localização condutor - lista de pontos da trajectóri ***Feedback*** - /
Homebase Channel	Requisitante	**Homebase do Requisitante**	***Publish*** - dados de utilizador - localização requisitante - destino do pedido - raio tolerância - duração do pedido ***Feedback*** - lista ofertas

Tabela A.3: Tabela de sumária de canais subscritos na solução 2

Canal	***Publishers***	***Subscribers***	**Eventos**
Requests Channel	Requisitante	**Homebase do Condutor**; Filtro: Lista de pontos interpoladores da trajectória do condutor	***Publish*** - dados de utilizador - localização requisitante - destino do pedido - raio tolerância - duração do pedido ***Feedback*** - lista de ofertas
Homebase Channel	Condutor	**Homebase do Condutor**	***Publish*** - dados de utilizador - localização condutor - lista de pontos da trajectória ***Feedback*** - /
Homebase Channel	Requisitante	**Homebase do Requisitante**	***Publish*** - dados de utilizador - localização requisitante - destino do pedido - raio tolerância - duração do pedido ***Feedback*** - lista de ofertas

Tabela A.4: Tabela de sumária de canais subscritos na solução 3

Canal	***Publishers***	***Subscribers***	**Eventos**
Deals Channel	Requisitante	**Condutor**; Filtro: ID do condutor	***Publish*** - dados de utilizador - id da oferta - id do condutor ***Feedback*** - viabilidade (V | F) - dados pto encontro

Tabela A.5: Tabela de sumária de canais subscritos na fase de acordo de boleia

Bibliografia

[1] I. Armstrong Consulting. Dedicated short range communications (dsrc) home. `http://www.leearmstrong.com/DSRC/DSRCHomeset.htm`.

[2] A. Auvinen, M. Vapa, M. Weber, N. Kotilainen, and J. Vuori. Chedar: peer-to-peer middleware. *Parallel and Distributed Processing Symposium, International*, 0:252, 2006.

[3] A. V. Bakre and B. R. Badrinath. Handoff and systems support for indirect tcp/ip. In *MLICS '95: Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*, pages 11–24, Berkeley, CA, USA, 1995. USENIX Association.

[4] H. Balakrishnan and R. Katz. Explicit Loss Notification and Wireless Web Performance. In *IEEE GLOBECOM Global Interne*, Sydney, Australia, November 1998.

[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving tcp performance over wireless links. *IEEE/ACM Trans. Netw.*, 5(6):756–769, 1997.

[6] J. Barreto, P. Ferreira, and M. Shapiro. Exploiting our computational surroundings for better mobile collaboration. pages 110–117, May 2007.

[7] K. Brown and S. Singh. M-tcp: Tcp for mobile cellular networks. *SIGCOMM Comput. Commun. Rev.*, 27(5):19–43, 1997.

[8] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. pages 117–134, 2006.

[9] V. Bychkovsky, K. Chen, H. Balakrishnan, and S. Madden. Cafnet: Carry-and-forward networking. `http://cartel.csail.mit.edu/cafnet.html`.

[10] R. Caceres and L. Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal on Selected Areas in Communications*, 13:850–857, 1994.

[11] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 233–245, New York, NY, USA, 2005. ACM.

[12] Y.-h. Chu, S. G. Rao, and H. Zhang. A case for end system multicast (keynote address). In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12, New York, NY, USA, 2000. ACM.

[13] D. D. Clark. The structuring of systems using upcalls. In *SOSP '85: Proceedings of the tenth ACM symposium on Operating systems principles*, pages 171–180, New York, NY, USA, 1985. ACM.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms - Section 24.3: Dijkstra's algorithm*. The MIT Press, 2nd revised edition edition, September 2001.

[15] P. C. Dillinger and P. Manolios. Bloom filters in probabilistic verification. In *In Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD*, pages 367–381. Springer-Verlag, 2004.

[16] S. Duarte. *DEEDS - A Distributed and Extensible Event Dissemination Service. PhD thesis*. Departamento de Informática FCT/UNL, 09 2005.

[17] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: vehicular content delivery using wifi. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 199–210, New York, NY, USA, 2008. ACM.

[18] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 29–39, New York, NY, USA, 2008. ACM.

[19] K. Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA, 2003. ACM.

[20] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt. Micro-blog: sharing and querying content through mobile phones and social participation. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 174–186, New York, NY, USA, 2008. ACM.

[21] Google. Google maps. `http://maps.google.com/`.

[22] W. Grosky, A. Kansal, S. Nath, J. Liu, and F. Zhao. Senseweb: An infrastructure for shared sensing. *Multimedia, IEEE*, 14(4):8–13, Oct.-Dec. 2007.

[23] S. B. E. A. T. C. Hong Lu, Nicholas D. Lane. Bubble-sensing: A new paradigm for binding a sensing task to the physical world using mobile phones. In *Proc. of International Workshop on Mobile Devices and Urban Sensing*, Apr. 2008.

[24] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138, New York, NY, USA, 2006. ACM.

[25] D. Jiang and L. Delgrossi. Ieee 802.11p: Towards an international standard for wireless access in vehicular environments. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2036–2040, 2008.

[26] M. Kim, J. W. Lee, Y. J. Lee, and J.-C. Ryou. Cosmos: A middleware for integrated data processing over heterogeneous sensor networks. *ETRI*, vol.30(no.5):pp.696–706, Oct. 2008.

[27] N. Kotilainen, M. Weber, M. Vapa, and J. Vuori. Mobile chedar "a peer-to-peer middleware for mobile devices. In *PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 86–90, Washington, DC, USA, 2005. IEEE Computer Society.

[28] U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi. Mobeyes: smart mobs for urban monitoring with a vehicular sensor network. *IEEE Wireless Communications*, 13:52, 2006.

[29] Microsoft. Senseweb. `http://research.microsoft.com/en-us/projects/senseweb/`.

[30] R. Morris, J. Jannotti, F. Kaashoek, J. Li, and D. Decouto. Carnet: A scalable ad hoc wireless network system. In *In Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System*, pages 61–65. ACM Press, 2000.

[31] F. Neves. *Suporte à Cooperação em Computação Móvel e Ubíqua. MSc thesis*. Departamento de Informática FCT/UNL, 11 2009.

[32] OpenStreetMapCommunity. Java openstreetmap editor - http://josm.openstreetmap.de/.

[33] OpenStreetMapCommunity. Openstreetmap - http://www.openstreetmap.org/.

[34] J. Scott, J. Crowcroft, P. Hui, and C. Diot. Haggle: a Networking Architecture Designed Around Mobile Users. In *WONS 2006 : Third Annual Conference on Wireless On-demand Network Systems and Services*, pages 78–86, January 2006.

[35] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. pages 30–41, 2003.

[36] Y. Zhang, B. Hull, I. Balakrishnan, and S. Madden. Icedb: Intermittently-connected continuous query processing. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 166–175, 2007.