Pontifícia Universidade Católica
do Rio de Janeiro

**André Victor Gomes de Aboim Mac Dowell**

# Uma API para Exergames Móveis com eventos centrados em microlocalização baseada em BLE Fingerprinting



**Dissertação de Mestrado**

Dissertação apresentada ao Programa de Pós-graduação em Informática da PUC-Rio como requisito parcial para obtenção do título de Mestre em Informática.

Orientador: Prof. Markus Endler

Rio de Janeiro
Fevereiro de 2016

Pontifícia Universidade Católica
do Rio de Janeiro

**André Victor Gomes de Aboim Mac Dowell**

# Uma API para Exergames Móveis com eventos centrados em microlocalização baseada em BLE Fingerprinting

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

**Prof. Markus Endler**
Orientador
Departamento de Informática – PUC-Rio

**Prof. Noemi Rodriguez**
Departamento de Informática – PUC-Rio

**Prof. Luis Valente**
Departamento de Ciência de Computação – UFF

**Prof. Márcio da Silveira Carvalho**
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 04 de fevereiro de 2016

**André Victor Gomes de Aboim Mac Dowell**

Graduou-se em Engenharia de Computação pela Pontifícia Universidade Católica em 2013. Bolsista CNPq de 2013 a 2015.



Ficha Catalográfica

MacDowell, André Victor Gomes de Aboim

Uma API para exergames móveis com eventos centrados em microlocalização baseada em BLE fingerprinting / André Victor Gomes de Aboim Mac Dowell ; orientador: Markus Endler. – 2016.
106 f. : il. color. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2016.
Inclui bibliografia

1. Informática – Teses. 2. Jogos pervasivos. 3. Exergames móveis. 4. Fingerprinting. 5. Redes de área pessoal sem fio. 6. Serviços baseados em localização. I. Endler, Markus. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

## Agradecimentos

À minha família e amigos, por todo o apoio.

Ao meu orientador, prof. Markus Endler, pela paciência, incentivo e pelo conhecimento, obrigado.

Aos meus colegas do LAC e do Telemídia, em especial ao Marcos Roriz, Luis Talavera e Álan Livio pelo constante apoio, ajuda e incentivo.

Aos meus colegas da Kimeric Labs e da Stone, os quais sempre me apoiaram nos momentos difíceis.

A minha namorada Bruna Queiroz, pela compreensão e por estar do meu lado em todas as minhas decisões.

Ao LAC, à CNPq e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

# Resumo

Mac Dowell, Andre Victor Gomes de Aboim; Endler, Markus. **Uma API para Exergames Móveis com eventos centrados em microlocalização baseada em BLE Fingerprinting.** Rio de Janeiro, 2016. 106p. Dissertação de Mestrado – Departamento de Informático, Pontifícia Universidade Católica do Rio de Janeiro.

Smartphones estão cada vez mais presentes no dia a dia de nossa sociedade, seja para trabalho ou lazer. Nessa plataforma móvel, há um crescente número de jogos eletrônicos que usam os sensores e funcionalidades disponíveis no celular para possibilitar sua jogabilidade, como uso de GPS para Jogos Baseados em Localização, uma categoria de Jogos Pervasivos Móveis. Por outro lado, existem outras categorias de jogos pervasivos que requerem funcionalidades especificas que o celular não dispõe diretamente, como detecção de proximidade entre usuários e uma localização mais precisa, rápida e disponível que o GPS. Adicionalmente, tecnologias de sensoriamento e de beacons para Internet das Coisas (IoT) estão cada vez mais baratas e disponíveis, estas que são muito usadas em soluções de microlocalização e posicionamento em diferentes contextos de aplicação. Em Exergames Móveis, uma categoria de jogos pervasivos onde a jogabilidade ocorre em um ambiente outdoor e tem ritmo de jogabilidade acelerado e com constante interação entre múltiplos jogadores, uma solução de localização mais precisa que o GPS vê-se necessária. O desenvolvimento desses jogos inclui execução de sessões de jogo e de seus componentes, assim como interoperabilidade entre diferentes tecnologias. Dessa forma, nesse trabalho, apresentamos como contribuições uma estratégia de localização usando Fingerprinting com beacons Bluetooth LE (BLE) e uma API para requisições e eventos comuns de localização. Analisamos a estratégia de localização em testes com diferentes configurações, através do uso de um Middleware para Jogos Pervasivos com gerência de Sessões, e avaliamos a API de localização com abstrações da jogabilidade de diferentes jogos pervasivos.

## Palavras-chave

Jogos Pervasivos; Exergames móveis; Fingerprinting; Redes de Área Pessoal sem fio; Serviços baseados em localização; Bluetooth Low Energy.

## Abstract

Mac Dowell, Andre Victor Gomes de Aboim; Endler, Markus (Advisor). **An API for Mobile Exergames microlocalization events using BLE Fingerprinting.** Rio de Janeiro, 2016. 106p. MSc. Dissertation - Departamento de Informático, Pontifícia Universidade Católica do Rio de Janeiro.

Smartphones are ever more present in the day to day lives of our society, for both work and entertainment. In this mobile platform, there is a growing number of games that uses sensing capabilities of the Smartphone for its gameplay mechanics, like GPS for location-based games, a category of mobile pervasive games. Although, there are categories of pervasive games that require specific hardware capabilities not normally found in a Smartphone, like precise proximity inference between devices and a more precise, fast and reliable location solution then GPS. Simultaneously, both sensing and beacon technologies for Internet of Things (IoT) are getting cheaper and more available, and there are many micro-locating solutions that uses these technologies in different application contexts. In Mobile Exergames, a category of pervasive games where the gameplay is outdoors and fast-paced, with constant interaction between multiple players, a more precise location solution then GPS is necessary. The development of these games includes the execution of game sessions and their components, together with the interoperability of different technologies. That way, in this work, we present a location strategy using Fingerprinting and Bluetooth LE (BLE) beacons and a API for common location requests and events. We analyze the location strategy through tests with different configurations using a Pervasive Game Middleware with Session management, and evaluate the location API through gameplay abstractions for a few pervasive games.



## Keywords

Pervasive Games; Mobile Exergames; Fingerprinting; Wireless Personal Area Network; Location-based services; Bluetooth Low Energy.

## Sumário

## Lista de figuras

## Lista de tabelas

*What is a master but a master student? And if that's true, then there's a responsibility on you to keep getting better and to explore avenues of your profession.*

Neil Peart

# 1
# Introdução

Este capítulo apresenta a contextualização do problema investigado e a motivação por trás do trabalho, além de um sumário dos desafios encontrados e objetivos. Em seguida, é apresentada uma descrição da organização da dissertação.

## 1.1.
## Contextualização

Recentemente, tem-se visto um aumento na disponibilidade e na acessibilidade de tecnologias de *Internet das Coisas* (Internet of Things ou IoT), como descrito por diferentes fontes [1, 2].

Esse fato é acentuado pela crescente integração dessas tecnologias com aparelhos móveis e computadores pessoais, graças a investimentos de grandes empresas de tecnologia em soluções próprias, como Intel [3] e Cisco [4], a interfaces comuns de comunicação sem fio como Bluetooth Low Energy (BLE) [24], RFID tags e infravermelho, e a crescente pervasividade de plataformas comuns de desenvolvimento, como os sistemas operacionais móveis Android [25] da Google e iOS [26] da Apple, e ambientes de prototipagem próprios como Arduino [27], Raspberry Pi [28] e ARM Cortex [29]. A diferença dessas novas soluções de outras anteriores é que estas são plataformas comuns de desenvolvimento que não visam somente a desenvolvedores [5, 6].

De acordo com o Grupo Gartner [7], existem referências do termo IoT desde o início da década de 90, e com elas, diferentes definições. Uma delas diz que:

> *"A Internet das Coisas representa uma rede de objetos físicos que incorpora tecnologias de comunicação e sensoriamento capazes de recuperar o estado interno desses objetos e do ambiente em sua volta."* [7]

Atzori et al. [8] incorpora IoT no paradigma de *Redes de Sensores sem Fio* (Wireless Sensor Networks ou WSNs), citando exemplos de *Objetos Inteligentes*

(Smart Objects, SOs) capazes de comunicação sem fio como RFID Tags, sensores, atuadores e Smartphones em aplicações complexas. O uso desses objetos possibilita o desenvolvimento de soluções em IoT para WSNs em diversas áreas como saúde e acessibilidade [34], movimentação urbana [35, 36], localização indoor [33], casas inteligentes [34, 37] entre outros.

Há também a crescente pervasividade de *Smartphones* no dia a dia da sociedade. De acordo com Pew Research Center [9], 64% dos cidadãos adultos americanos e 84% dos jovens possuem Smartphones (dados do final de 2014), sendo que quase 90% desses usuários se conectam à Internet com esses Smartphones. Outro artigo [56] aponta que, em 2016, é esperado que mais de 80% dos usuários de Smartphones terão algum jogo instalado em seus aparelhos, o que significa que uma parcela significativa da população estará interessada em jogos móveis.

Neste contexto, em que o desenvolvimento de soluções que usam tecnologias IoT é facilitado por sua crescente acessibilidade e barateamento [30] e que se espera que a maioria da população esteja conectada à Internet através do uso de Smartphones e interessada em jogos, há uma oportunidade de desenvolvimento e pesquisa de soluções em uma área inovadora de jogos chamada de *Jogos Pervasivos Móveis.*



## 1.2. Motivação e Desafios

A pergunta que iniciou o interesse por esta pesquisa foi “Quais são os desafios de desenvolvimento e execução de Jogos Pervasivos relacionados à IoT? ”. A definição de Jogos Pervasivos, como descrito no trabalho de Nieuwdorp [10], é variável e depende da perspectiva ou contexto, podendo ser tecnológica ou cultural. No contexto tecnológico, significa todo o jogo eletrônico que utiliza o ferramental de tecnologias de *computação pervasiva* para a execução do jogo. No contexto cultural, que foca em como os componentes virtuais do jogo se relacionam com o mundo real, é todo jogo eletrônico em que os jogadores, personagens, lugares e o enredo do jogo se permeiam e se misturam com o mundo real. Montola [40] também diferencia jogos pervasivos *capacitados por tecnologia* e jogos *sustentados por tecnologia*, onde este último trata de jogos que não podem existir caso não tenham os requisitos tecnológicos necessários.

Benford et al. [11], Broll et al. [12] e Walther [13] citam os desafios tecnológicos comuns ao desenvolvimento e execução de jogos pervasivos com diferentes conceitos e objetivos.

Benford et al. [11] listam como desafios as incertezas relacionadas a comunicação móvel e sem fio; arquiteturas híbridas em sistemas distribuídos de cliente-servidor (definição de protocolos de comunicação); mapeamento dos elementos do jogo do ambiente virtual para o mundo real; configuração do ambiente de jogo (local físico), jogadores, sessão e elementos independentes do jogo (sensores e outros objetos físicos); e orquestração do jogo, que diz respeito à operação e gerenciamento do jogo em tempo real.

Neste trabalho, consideramos que estes desafios tecnológicos, em conjunto com a manipulação dos estados dos objetos físicos do jogo, dos jogadores e às condições de vitória do jogo, assim como as soluções para comunicação entre os jogadores e sua localização, constituem nos desafios reais para a *execução plena* de um jogo pervasivo.

Broll et al. [12] categorizam os problemas de desenvolvimento de jogos pervasivos como aqueles relacionados a localização em tempo real dos elementos e jogadores do jogo (sendo esse o grande desafio inerente a todos os jogos pervasivos baseados em localização); comunicação entre jogadores e o servidor de jogo; desafios relacionados a transmídia para jogos de realidade aumentada; integração e implantação de hardware específico a ser usado, seja aparelhos pessoais como Smartphones ou aparelhos espalhados pelo ambiente como beacons, sensores, atuadores e outros objetos físicos; uso de motores (*engines*) de jogo; e orquestração e gerenciamento em tempo real do jogo.

Já Walther [13] lista quatro elementos como ‘eixos’ para o desenvolvimento de jogos pervasivos: (i) Distribuição, no que diz respeito no conjunto de tecnologias de comunicação, sensoriamento e hardware específico necessário para executar o jogo; (ii) Mobilidade, no que diz respeito a acompanhamento dos jogadores e objetos móveis do jogo, assim como problemas tecnológicos referentes a sistemas de localização em tempo real, quando aplicáveis; (iii) Persistência, tratando-se dos desafios para manter o jogo em execução e funcional por tempo indeterminado ou determinado; (iv) Transmídia.

Analisando os desafios tecnológicos descritos até aqui, foi determinado que este trabalho exploraria questões referentes a *localização em tempo real* dos

jogadores em jogos pervasivos. Os desafios de localização são relacionados à obtenção de informação de posicionamento dos jogadores do jogo em tempo real e de forma precisa e confiável, tópicos comumente discutidos em trabalhos de para *Jogos Baseados em Localização* (LBGs).

Consideraremos que os demais desafios citados para a execução plena de um jogo pervasivo, como a execução da sessão de jogo, gerenciamento dos recursos e manuseio e processamento dos dados e estados de todos os elementos do jogo, e a interoperabilidade e comunicação entre as tecnologias usadas no jogo, podem ser tratados por um Middleware específico para a execução desses Jogos e que *não tem implicação direta na solução de localização*. Estes desafios de execução tem uma relação direta com questões relacionadas à comunicação entre componentes de sistemas distribuídos, assim como desafios tecnológicos de IoT.

Levando em consideração os tipos de jogos que poderiam apresentar maiores desafios relacionados à orquestração e localização, buscou-se definições de subgêneros ou categorias de Jogos Pervasivos. Alguns autores como Nieuwdorp [10], Walther [13] e Magerkurth et al. [14] buscam categoriza-los de diferentes formas, separando-os em subgêneros através das diferentes tecnologias usadas, aspectos sociais envolvidos e fisicalidade das interações, por exemplo. Em alguns casos, um jogo se enquadra em um subgênero ou outro unicamente sob a perspectiva da tecnologia empregada para sua execução, como comenta Nieuwdorp [10].

Buscando relação entre fisicalidade e socialização em jogos pervasivos, temos a definição de *Exergames Móveis*, de Wylie et al. [15], Chittaro et al. [16] e Macvean et al. [17], que os consideram jogos eletrônicos que combinam exercícios físicos em sua jogabilidade e usam aparelhos móveis, como Smartphones, como plataforma capacitadora, além de ter como foco a interação entre múltiplos jogadores. Estes jogos têm ritmo de jogabilidade acelerado, onde jogadores podem se movimentar dentro de uma área delimitada, muitas vezes simulando jogos, brincadeiras e esportes praticados no mundo real. Görgü et al. [18] vão ainda um pouco além, definindo *Freegaming* como sendo jogos pervasivos com mobilidade (como LBGs), com tecnologia de realidade aumentada, colaborativos (com suporte a multijogadores) e adaptativos (capazes de se adaptar a um ambiente outdoor mutável).

Independente da nomenclatura, os termos que sobressaem na definição de exergames móveis e na relação com outros tipos de jogos pervasivos são dois: *interatividade entre múltiplos jogadores* e *mobilidade*. Interatividade, trazendo tanto o aspecto social quanto físico, pois estes jogos têm sua jogabilidade construída baseada na interação entre jogadores e outros componentes de jogo, de forma competitiva ou colaborativa; e mobilidade, pois os jogadores se movimentam em uma localidade física no mundo real onde o jogo está ocorrendo. Assim, como descrito por Wylie et al. [15], Chittaro et al. [16] e Macvean et al. [17], vemos que Smartphones de fato podem ser uma plataforma capaz de trazer soluções para estes dois conceitos.

Considerando os subgêneros de jogos pervasivos, as definições de exergames móveis e os desafios relacionados à localização e execução desses tipos de jogos, foi reescrita a questão de pesquisa para: "Como resolver os desafios de localização com precisão fina em jogos pervasivos no contexto de um exergame móvel? ".

Existem alguns exemplos de LBGs lançados publicamente que fazem uso do Smartphone, e em que a solução de localização está baseada no uso do GPS interno dos aparelhos. Jogos como o Ingress [19], da Google, "Zombies, Run! " [20] e Geocatching [21] são exemplos conhecidos de LBGs. Esses jogos têm como característica a pervasividade de espaço (ocorrem/podem ocorrer em qualquer lugar do mundo) e tempo (ocorrem/podem ocorrer a qualquer momento). Além disso, dados de sensores internos do Smartphone (GPS e acelerômetro por exemplo) são coletados de forma automática pelo smartphone do jogador. Sua solução de localização através do GPS é suficiente já que não há necessidade para uma precisão maior que a disponível pelo GPS tradicional, que é comumente de 7 a 15 metros, podendo chegar a 4 metros [22, 23].

Quando tratamos de LBGs que lidam com contato físico, ou necessidade de identificação de proximidade entre jogadores mais precisa, como o Human Pacman [38] e o Hot Potato [39], o uso de GPS tradicional como uma solução de localização passa a ser um problema, pois a precisão necessária nesses casos é maior do que o limite disponível do GPS e não há garantias que o GPS estará disponível a todos os momentos durante o jogo. Também, em jogos como o Human Pacman e o Hot Potato, as soluções para jogabilidade vem de hardware proprietário (uma mochila com um notebook, visor, bateria, sensor de toque entre

outros no caso de Human Pacman, e aparelhos Sun SPOT no caso de Hot Potato), e não de uma solução de hardware padronizada e acessível como são os Smartphones. Outros exemplos de exergames móveis serão apresentados e analisados no Capítulo 3.

No trabalho de Rashid et al. [31], são analisados sistemas de localização que podem ser usados para LBGs executando em Smartphones. GPS é citado como sendo uma das soluções principais dado o seu alto nível de integração com os celulares (2006 na época, hoje em dia sendo lugar-comum para Smartphones), tendo como principal problema as restrições em meios urbanos, ambientes *indoor* e latência na aquisição da localização. GPS assistido (A-GPS) [86] é descrito como uma evolução natural do GPS tradicional, que diminui o tempo de aquisição e aumenta a precisão através de informações recebidas da rede celular através de estações base de transmissão. Porém, problemas relacionados à interferência de múltiplos caminhos (*multipath*[1]) ainda ocorrem em A-GPS, o que impede que a medição seja mais precisa. Por último, são introduzidas soluções de localização implícita, onde o celular interage com outros objetos e sistemas que têm uma localização conhecida de modo a ter sua própria localização aproximada deduzida. Soluções de localização por *intensidade de sinal de radiofrequência* (RSSI) se enquadram nessa categoria. Trabalhos que aplicam essas soluções serão discutidos nas seções seguintes.

Podemos traçar um paralelo entre a definição acolhida de exergame móveis, sendo um jogo com interação quase-física entre múltiplos jogadores, com a descrita no trabalho de Akribopoulos [39, pag. 3], que o define como um LBG de ritmo acelerado e com contato físico entre múltiplos jogadores. . Isso significa que uma solução diferente do uso do GPS integrado ao Smartphone é necessária para o sistema de localização dos participantes do jogo, já que é necessária a identificação da localização e distâncias relativas entre os jogadores a qualquer momento durante o jogo.

A proposta de uma solução para esses desafios, relacionados às dificuldades em se obter localização mais precisa e disponível que o GPS para exergames móveis, é o tema central desse trabalho. Para a solução de localização, como Smartphones são as plataformas capacitadoras para os exergames móveis, foi

[1] Ocorre quando uma determinada onda chega a um receptor através de múltiplos caminhos devido a reflexão de outros componentes externos.

proposta uma API de requisições e eventos de localização de jogadores que usa localização implícita através da demarcação de regiões do local físico do jogo com sensores, de modo possibilitar localização por RSSI quando o jogador se encontra em uma dessas regiões, e GPS tradicional quando não se encontra. Para o estudo de caso da solução, é apresentado um Middleware para Jogos Pervasivos capaz de criar e gerenciar a execução de sessões de jogos com múltiplos jogadores.

## 1.3. Objetivos

Nesse trabalho, buscamos propor soluções para os problemas de localização encontrados no desenvolvimento de exergames móveis. A solução proposta é uma *API de requisições e eventos de localização* de jogadores para exergames móveis.

Segue, assim, uma enumeração dos objetivos do trabalho:

1. Apresentar um Middleware para exergames móveis, capaz de lidar com configuração e execução de sessões exergames móveis com definições de área de jogo, sub-regiões com características distintas e número de jogadores – Capítulo 4;
2. Propor um método baseado em busca de localização implícita usando RSSI para localização dos jogadores no ambiente de um exergame móvel – Capítulo 5;
3. Analisar o método de localização por RSSI propostos através de testes de uma prova de conceito em diferentes configurações e condições – Capítulo 5;
4. Propor e implementar uma API de requisições e eventos de localização que usa o método de localização por RSSI com Smartphones, possibilitando assim o controle e acompanhamento de diferentes informações de localização e distância entre jogadores – Capítulo 6;
5. Mapear os eventos de localização e jogabilidade de alguns exemplos de exergames pervasivos usando a API proposta – Capítulo 6.

## 1.4. Organização da Dissertação

Esta dissertação é organizada da seguinte forma:

- Capítulo 1 - A introdução, onde é apresentado o contexto tecnológico do trabalho realizado, as motivações, a pesquisa feita para identificar a questão de pesquisa, os desafios, objetivos e a organização da dissertação.

- Capítulo 2 - Neste capítulo, são abordados os conceitos gerais de Jogos Pervasivos e o conceito tecnológico e cultural de Exergames Móveis usado nesse trabalho. Também são definidos termos comumente usados ao longo do texto.

- Capítulo 3 – Neste capítulo, são apresentados exemplos de exergames móveis e suas características, assim como comentários a respeito de seu funcionamento e implementação.

- Capítulo 4 – Este é o capítulo em que é introduzido o Middleware de Exergame Móveis. Através dele, é possível lidar com a configuração e execução de exergames móveis, criação e configuração de sessões de jogo, incluindo o ambiente físico (área do jogo e sub-regiões de microlocalização) e o contexto de execução (número de jogadores, duração, divisão de times, uso de sensores e eventos da API de localização). Assim, com o uso deste, é possível pôr em prática a análise da prova de conceito apresentada no Capítulo 5. Inicialmente são apresentados alguns trabalhos relacionados à frameworks e propostas de ambiente padronizados para jogo pervasivos e, em seguida, a arquitetura geral do middleware e a implementação proposta.

- Capítulo 5 – Neste capítulo, é introduzida a estratégia de localização implícita usada para a localização baseada em RSSI desse trabalho: Fingerprinting por amostragem, como prova de conceito para a API de Localização. Inicialmente, são apresentados trabalhos relacionados à diferentes tipos de sistemas de localização implícita, assim como o conceito de fingerprinting. Em seguida, o conceito de fingerprinting por amostragem e suas fases é aprofundado. Depois, a estratégia usada no contexto de um exergame móvel é descrita e a solução implementada usando beacons *Bluetooth Low Energy* (BLE) [32] é apresentada. Seguindo, é apresentado a forma como a solução proposta é integrada com o middleware e usada pela API de requisições e eventos. Por último, os testes feitos no contexto de jogo, resultados, problemas da abordagem e melhorias futuras são apresentados nessa ordem.

- Capítulo 6 – O último capítulo de contribuição deste trabalho é acerca da API de requisições e eventos de localização proposta. O conceito de uma API SOA dirigida a eventos é apresentado, assim como trabalhos relacionados. Em seguida,

a arquitetura e os elementos da API são apresentados e descritos, assim como a implementação da mesma para o contexto de jogo. Por último, são demonstradas abstrações da jogabilidade de alguns dos exergames móveis apresentados anteriormente no Capítulo 3 usando a API, de forma a avaliar o uso da mesma e problemas de abordagem e melhorias futuras são discutidos.

- Capítulo 7 – Este capítulo final apresenta as conclusões e análise do trabalho realizado, passando pela implementação do middleware, a estratégia usada e os resultados obtidos dos testes de fingerprinting por amostragem e as abstrações dos exemplos de exergames móveis usando a API de requisições e eventos. Em seguida, são apresentados alguns possíveis trabalhos futuros e os problemas que eles podem abordar.

# 2
# Fundamentação Conceitual

Este capítulo apresenta os conceitos necessários para definir o contexto dos desafios que esse trabalho visa atacar. Para isso, buscamos primeiro definir jogos pervasivos no contexto geral mais aceito pela comunidade acadêmica para, em seguida, definirmos exergames móveis como um subgênero de jogos pervasivos com características bem distintas. Além disso, comentaremos o contexto tecnológico esperado para os exergames móveis visados por esse trabalho e listaremos alguns termos que são usados ao longo do trabalho.

## 2.1.
## Jogos Pervasivos

Como Valente et al [60] discutem em seu trabalho, ainda não há um consenso a respeito da definição de jogos pervasivos na comunidade acadêmica, pois o termo ainda é relativamente recente. Nieuwdorp [10] busca listar subgêneros de jogos de forma a tentar definir Jogos Pervasivos como sendo "*a categoria de jogos que conecta vários subgêneros com contextos tecnológicos e culturais distintos, misturando-os no mundo real*". Montola [41], Benford et al. [11] e Magerkurth et al. [14] definem, em seus trabalhos, o termo de uma forma mais direta como sendo "*jogos eletrônicos que expandem a experiência de jogo para o mundo real*", ou "*jogos eletrônicos em que as dimensões espaciais, temporais e sociais se expandem para o mundo real*".

Valente et al [60] vão um pouco além, definindo jogos pervasivos móveis como sendo "*a categoria de jogos pervasivos em que Smartphones são usados como tecnologia capacitadora para ciência de contexto*". Este trabalho vai de acordo com a definição de Valente et al quando consideramos o subgênero exergame móveis como sendo do mesmo contexto tecnológico que jogos pervasivos móveis.

## 2.2.
## Exergames móveis

Exergames são uma categoria de jogos eletrônicos que nem sempre é diretamente vinculada a jogos pervasivos. Em um dos primeiros trabalhos sobre o tema, Mueller et al. [87] definem Exertion Interfaces como interfaces capazes de deliberadamente requerem o uso de esforço físico, possibilitando um jogo de esporte à distância [88]. O contínuo trabalho de Mueller trouxe a comunidade vários outros exemplos de "Jogos de Exaustão" nos anos seguintes [89, 90]. Já Sinclair [42] define *exergaming* simplesmente como sendo "*o uso de jogos eletrônicos no contexto de uma atividade física*", citando jogos como Wii Fit [43] e Just Dance [44], e a difusão de tecnologias de movimento para consoles de mesa como Wii Mote [45], Kinect [46] e Playstation Move [47] como exemplos de exergaming.

Existem definições que se aproximam mais do contexto desse trabalho, como a de Laine [48], que define *Exergames Pervasivos Distribuídos* como "*exergames que combinam tecnologias de sistemas distribuídos, sensores, e tecnologias de ciência de contexto em um contexto cultural de desafios físicos*". Já Chittaro [16], Wylie [15] e Görgü [18] definem *Exergames Móveis* como sendo "*exergames capacitados por um aparelho móvel carregado pelo jogador*", apresentando-o como um subgênero de LBGs, porém no contexto de um jogo que encoraja exercícios físicos e interação entre múltiplos jogadores.

O trabalho de Wylie, de 2008, é o mais antigo a definir exergames móveis no contexto buscado, tratando-o como uma "*categoria de jogos pervasivos baseados em localização, capacitados pelo uso de Smartphones, com foco em interatividade entre múltiplos jogadores e com o objetivo de encorajar exercício físico*". Porém, o trabalho apresentado por Cheok et al. [38], de 2004, em que ele apresenta o jogo Human Pacman se enquadra no contexto cultural de exergames móveis, sendo pioneiro não somente na área de exergames, como também na área de jogos pervasivos antes mesmo desses termos serem amplamente difundidos:

> *"Acreditamos que Human Pacman é pioneiro em uma nova forma de jogo que foca em contato físico, mobilidade, interação social e computação ubíqua. "* Cheok et al. [38]

Assim, tratando-se do *contexto cultural*, foi definido que jogos pervasivos que tem como máximas em sua jogabilidade a *interatividade* (entre jogadores ou entre o jogador e seu Smartphone ou outros elementos físicos do jogo) e *mobilidade* (em conjunto com exercício físico) podem ser classificados como exergames móveis nesse contexto. Além disso, como comentado por Akribopoulos [39], jogos desse tipo têm as interações entre os jogadores como sendo físicas ou próximas de físicas (simuladas ou não) e a mobilidade dos jogadores podem ter a característica de serem de ritmo acelerado.

Adicionalmente, considerando as definições dos trabalhos encontrados, foi decidido que o *contexto tecnológico* de um exergame móvel consistente com a definição usada nesse trabalho deve:

- Ter jogadores humanos carregando Smartphones e interagindo entre si (fisicamente ou não);

- Ocorrer em um local e tempo fixo, assim como um jogo esportivo ou de playground;

- Ter algum tipo de uso de sensores, sejam externos (para localização e/ou interação com objetos físicos) ou dos Smartphones (para interações entre jogadores e consigo mesmo);

- Ter uma entidade controladora no paradigma de sistemas distribuídos, dado que os jogadores carregam Smartphones (clientes móveis). Essa entidade, que pode ser um servidor para administração e/ou monitoramento, recebe informações dos Smartphones dos jogadores a respeito de seus estados e do ambiente, podendo fazer algum processamento ou tomada de decisão dependendo dos objetivos e regras do jogo.



## 2.3. Termos usados

Alguns termos e definições que são descritos e usados em diferentes seções neste trabalho são:

- *Contexto cultural VS tecnológico:* Trata-se do foco dado a uma determinada análise de uma categoria de jogo. Contexto tecnológico consiste nas tecnologias usadas na execução e monitoramento do jogo, assim como a arquitetura e ambiente em que o mesmo se encontra. Contexto cultural é tudo aquilo que define

o jogo independente da tecnologia empregada no mesmo, como objetivos, características de interação entre jogadores e regras.

- *Pervasividade de espaço e tempo*: Trata-se da descrição de como o jogo ocorre no mundo físico nos quesitos de tempo e espaço. O jogo pode começar e ocorrer o tempo todo durante tempo indeterminado ou pode ter início e duração fixa (ou determinada por uma condição de vitória). Tratando-se de espaço, o jogo pode ocorrer em um local fixo pré-determinado (chamado de área de jogo), ou pode ocorrer em um espaço absoluto (tipicamente em toda a área de uma cidade, país ou todo o mundo). Jogos com tempo e espaço determinados são chamados de *jogos de sessão* (ou de evento, como apresentado no trabalho de Valente et al. [60]). Esportes e jogos de playground (como pique-pega e pique-bandeira) são tipicamente jogos de sessão, pois têm duração e locais determinados.

- *Elementos de Jogo*: São os elementos físicos que compõem o jogo, no caso, os jogadores (elementos humanos), objetos físicos e virtuais (elementos não-humanos). Os estados dos elementos físicos, caso estes façam parte da jogabilidade do jogo de maneira direta, interagindo com os jogadores, podem ser mapeados virtualmente de maneira automática caso sejam capacitados tecnologicamente para tal (sendo sensores que captam informações do ambiente e/ou interagem com o Smartphone do jogador, por exemplo). Objetos unicamente virtuais não tem necessidade de mapeamento. Personagens não-jogadores (NPCs) podem existir dependendo do contexto de jogo, podendo ser unicamente virtuais, caracterizados fisicamente por um objeto (um manequim, por exemplo) ou até mesmo interpretados por outros humanos fora do jogo (chamados atores).

- *Sessão de Jogo*: É o momento no tempo e espaço em que ocorre o jogo, juntamente com suas características. Uma partida de futebol, por exemplo, é uma sessão de jogo que ocorre em um local específico (estádio), tempo e duração (horário da partida e 90 minutos), com características de time (dois times, 11 jogadores em cada lado) e regras e condições de vitória distintas (após o término do tempo, vence quem tiver feito mais gols).

- *Localização relativa VS absoluta*: É dito localização relativa quando a localização de um jogador ou elemento do jogo é descrita em termos relativos a outro elemento posicionado absolutamente no jogo. Por exemplo, o jogador se encontra a 2 metros do centro de uma sub-região circular do jogo centrada em um ponto de latitude/longitude ‘a1, b1’, ou no quadrante ‘x, y’ de uma sub-região

centrada em ‘a2, b2’. Já localização absoluta é simplesmente a localização descrita pelo padrão absoluto definido para sistema, no caso, uma tupla latitude/longitude. Na Figura 2.1, os jogadores na sub-região vermelha estão em diferentes quadrantes na sub-região, esta que está posicionada absolutamente em uma coordenada latitude/longitude.

- *Área de jogo VS sub-regiões*: A área do jogo é definida como sendo o local físico do jogo. No exemplo de implementação do middlware apresentado nesse trabalho, podemos definir um ponto central com coordenada latitude/longitude e um raio em metros. Sub-regiões (podem haver várias) são regiões definidas dentro da área de jogo em que regras específicas distintas da região externa ocorrem. No caso do trabalho proposto, jogadores em sub-regiões são geolocalizados com microlocalização (localização relativa), enquanto jogadores fora de sub-regiões são geolocalizados normalmente (localização absoluta). Dependendo do jogo implementado, as regras de jogabilidade podem mudar se o jogador estiver dentro ou fora de uma sub-região (por exemplo, se ocorrer uma falta dentro da área do goleiro em uma partida de futebol, é pênalti, mas se ocorrer fora, é apenas falta). Na Figura 2.1, vê-se uma área de jogo com duas sub-regiões, uma vermelha e uma azul.

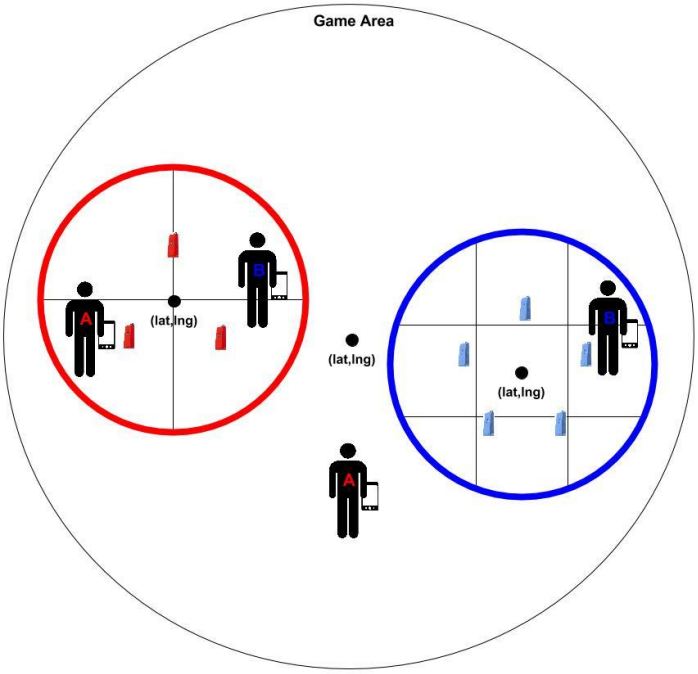


Figura 2.1: Exemplo de área de Jogo e suas sub-regiões. A sub-região vermelha tem quatro quadrantes e é demarcada por três beacons, enquanto a azul é de nove quadrantes e demarcada por cinco.

- *Microlocalização*: No contexto de um exergame móvel, trata-se da localização relativa de um jogador em uma sub-região do jogo. A forma que a localização foi adquirida não define ela ser uma microlocalização, e sim o fato dele ser relativa a uma sub-região (quadrante, radial, etc.) ao invés de absoluta (coordenada latitude/longitude). Na Figura 2.1, a posição nos quadrantes das sub-regiões é feita por localização implícita, enquanto fora das sub-regiões é realizada de forma absoluta (por GPS, por exemplo).

- *Beacons*: São os objetos físicos usados para mapear uma sub-região para prepará-la para microlocalização por método de localização implícita. São capazes de transmitir um sinal de radiofrequência de forma com que aparelhos próximos possam usar a intensidade do sinal recebido para a solução própria de localização implícita, caso desejado.

- *Orquestração*: Corresponde a supervisão, gestão e intervenção humanas no jogo – tanto antes da sessão (e.g. arrumar o local do jogo, instalar equipamentos) quanto em tempo real, durante uma sessão.

- Ritmo acelerado: Definição da velocidade das interações e de movimentos do jogador. A atualização dos dados do jogo, captura de informações de localização e processamento das condições deve escalar em conjunto com o ritmo acelerado do jogo, o que nem sempre é possível dada as escolhas tecnológicas de implementação para o jogo ou do Middleware usado para o mesmo. Para todos os efeitos, em exergames móveis, consideramos que o jogador tendo informações de localização e estados dos elementos do jogo atualizadas a cada segundo é suficiente para uma execução correta do jogo.

# 3
# Exemplos de Exergames Móveis

Neste capítulo, alguns jogos que se enquadram no contexto tecnológico e cultural de exergames móveis são descritos e analisados (exceto um, que é analisado apenas com o propósito de comparação). Os jogos são analisados em quesitos tecnológicos (tecnologia empregada para a execução, localização, comunicação entre os elementos do jogo, visualização) e culturais (objetivos, colaboração e competição entre jogadores, mobilidade dos jogadores, pervasividade em tempo e espaço). Alguns desses jogos não são diretamente classificados por seus autores como exergames móveis, ou sequer jogos pervasivos, porém, considerando as máximas de interatividade e mobilidade definidas para o contexto cultural de exergames móveis, foi considerado que esses jogos se enquadram nesse conceito (exceto quando denotado). Todos os jogos são trabalhos acadêmicos e/ou jogos voltados para demonstrações públicas em cidades ou eventos, exceto Botfighters [55, 57], que foi um jogo comercial em sua época.

Praticamente todas as tecnologias descritas como usadas nos jogos analisados são comparadas e classificadas no trabalho de Benford et al. [58 Pag. 8-10]. Um resumo dessas características tecnológicas e das culturais encontra-se na Tabela 3.1 ao final deste capítulo.



## 3.1.
## Exemplos de Jogos

*TimeWarp*

Em TimeWarp, de Herbst et al. [49], jogadores devem buscar elfos perdidos em sua cidade em diferentes épocas em um ambiente de realidade-mista (MR) enquanto resolvem desafios propostos por NPCs. Os jogadores carregam um aparelho portátil e um capacete para visualização AR. Além disso, o jogo é capaz de mapear em 3D a posição e orientação do jogador através de um sistema de rastreamento híbrido de seis graus de liberdade (6DOF). Através do aparelho portátil, os jogadores visualizam sua posição em um mapa e algumas informações de jogo.

A área do jogo é delimitada em uma região da cidade, porém, existem diferentes sub-regiões de jogo em que o jogador pode realizar diferentes ações: portais do tempo (para mudar de época), mercados (para comprar itens) e locais de desafio.

O jogo é um sistema distribuído e baseado em localização, onde informações do jogador como localização, itens, desafios vencidos e estados são guardadas pelo servidor de jogo. A localização do jogador é resolvida por GPS, exceto quando o mesmo entra em uma sub-região. Quando em uma sub-região, o capacete para AR é ligado e o jogador passa a ouvir sons referentes as ações que ele toma na região (entrar no portal, ou comprar um item, por exemplo).

Tecnologicamente, o jogo tem uma execução complexa, apesar da não necessidade da interação entre múltiplos jogadores, dada a multiplicidade dos elementos do jogo e a complexidade das ações e possibilidades de interação entre os jogadores e esses elementos. Os jogadores interagem com vários elementos virtuais, progredindo na história a medida que resolvem os desafios propostos. A comunicação entre os componentes é resolvida usando a Internet através de uma conexão Wi-Fi. A execução do jogo não requer uma orquestração por parte de terceiros.

Apesar de ser um jogo onde os jogadores não interagem diretamente entre si, TimeWarp é um exemplo interessante por ser um jogo de exploração urbana com localização absoluta e relativa, realidade mista, jogos em sessão e interação com objetos virtuais.



*NetAttack*

Neste jogo de Lindt et al. [53], incluindo alguns dos criadores de TimeWarp, os jogadores são divididos em dois times, cada um com um jogador fixo em um PC indoor e um ou mais jogadores outdoor. Os jogadores outdoor carregam consigo um capacete para visualização estereoscópica, uma mochila com um laptop e um aparelho de GPS, enquanto o jogador indoor consegue monitorar o que é filmado pelos jogadores do seu time e visualiza-los em um mapa. Os jogadores devem coletar artefatos espalhados por uma região antes dos jogadores do outro time. Os jogadores de times opostos devem se evitar, não podendo coletar um artefato caso um jogador de outro time esteja próximo.

Em NetAttack, os desenvolvedores tiveram que solucionar o problema de distância relativa entre jogadores assim como o proposto nesse trabalho, pois apesar da precisão do GPS ser suficiente para identificar se um jogador está numa região em que um artefato (fixo) se encontra, não é este o caso para distância relativa entre jogadores (móveis). A solução deles foi usar um sistema baseado em marcadores fixos através de um ARToolkit, como o visto na Figura 3.1.


Figura 3.1: Hiro Marker, um tipo de marcador de AR (Realidade Aumentada)

No contexto cultural, NetAttack se enquadra na categoria exergame móvel, pois apesar do objetivo direto não ser necessariamente que aja exercício físico, o mesmo acaba sendo uma consequência. Além disso, é um jogo outdoor, que requer colaboração entre um grupo de jogadores que se movimentam em uma área de jogo pré-definida. Se tratando da pervasividade de tempo e espaço, o jogo se enquadra em um jogo de sessão, pois ocorre em um local pré-definido e tem uma duração limitada (até os artefatos serem coletados).

No contexto tecnológico, o jogo também segue os paradigmas esperados. Sua execução é simples dado que a única informação coletada pelos jogadores que é processada de maneira automática é a sua localização, essa resolvida através de GPS tradicional (para apresentação em um mapa na tela do jogador indoor). A informação dos artefatos coletados é validada manualmente pelo jogador indoor (visualizando o artefato pela câmera estereoscópica) e a comunicação entre os componentes individuais é realizada pela Internet em uma rede Wi-Fi. A orquestração ocorre pelo preparo prévio da região de jogo, ao se espalhar os artefatos que os jogadores buscarão.

*Songs of North*

Em Songs of the North, de Lankoski et al. [50], o jogador interage com os elementos do jogo usando o teclado numérico de um celular em conjunto com movimentação em um ambiente de realidade mista. O mundo virtual, chamado de ‘mundo espiritual’ no trabalho, é executado em um servidor, este que mapeia as posições dos elementos virtuais e dos jogadores (localização por GPS tradicional) e lida com as regras do jogo. O jogo é persistente em tempo, ou seja, enquanto o servidor de jogo estiver executando-o, o mesmo não termina. O objetivo do jogo é encontrar sub-regiões na área de jogo em que ajam missões a serem resolvidas, missões essas que consistem em teclar um determinado padrão de teclas no teclado numérico do celular (este que simula um tambor). O jogo permite participação de multijogadores, e algumas missões só podem ser resolvidas caso aja mais de um jogador na sub-região da mesma. A comunicação entre jogadores é feita diretamente pelo celular, através de mensagens SMS, enquanto que os dados das ações dos jogadores e seu posicionamento é enviado para o servidor pela Internet.



*Viking Ghost Hunt*

Viking Ghost Hunt, de Carrigy et al. [51], é um jogo de narrativa dinâmica apoiado em localização, onde o jogador tem o papel de um investigador à procura de fantasmas. Como o jogo não é multijogador, a jogabilidade base consiste no jogador explorar a região de jogo (fixa, em volta de uma igreja em Dublin, Irlanda) completando missões e desbloqueando o andamento da história.

O jogador carrega um celular que serve como mapa e radar dos fantasmas na região. Ao se aproximar de um fantasma, o medidor mostrado na tela ficará agitado e o celular fará sons do fantasma próximo. Ao usar a câmera, o jogador poderá ver o fantasma e tirar uma foto dele, ‘capturando-o’. Os locais em que os fantasmas se encontram são fixos e a identificação da proximidade do jogador a esses locais é através de GPS tradicional. Essas regiões são propositalmente grandes, dado a falta de acurácia e lentidão da recuperação de localização do GPS. A apresentação do fantasma na câmera não usa nenhum objeto físico de AR, pois é simplesmente um modelo 3D posicionado em cima do stream de vídeo do celular.

Viking Ghost Hunt é um jogo apoiado a localização simples, dado seu uso do GPS tradicional unicamente para determinar a posição absoluta do jogador.

Sua execução ocorre toda no próprio celular e a necessidade de Internet através de uma conexão 3G ou Wi-Fi deve-se unicamente ao uso da API do Google Maps para apresentar o local que o jogador se encontra. Como ele ocorre somente para o jogador que usa o celular, sua duração é diretamente vinculada ao tempo que o jogador demora para completa-lo. Viking Ghost Hunt não se enquadra no contexto de um exergame móvel, apesar de ser um jogo baseado em localização, outdoor e com interação do jogador com o meio virtual à sua volta através do uso de um Smartphone. A escolha de apresenta-lo nessa seção foi para ter um exemplo de um jogo com características próxima das desejadas, mas que não chega a ser considerado um exergame móvel.

*Hot Potato*

Neste jogo apresentado no trabalho de Akribopoulos et al. [39], jogadores carregam aparelhos portáteis que simulam o ato de carregar uma batata. O objetivo do jogo é arremessar a batata para outros jogadores próximos antes que ela 'exploda'. As batatas são geradas aleatoriamente de tempos em tempos nos aparelhos dos jogadores que ainda estão participando do jogo. O jogo acaba quando somente um jogador restar, que é o vencedor. Jogadores que se afastam da área de jogo propositalmente tem uma chance maior de gerarem uma batata prestes a explodir.

O jogo é executado no framework apresentado no trabalho, chamado de FinN – Fun in Numbers. Este framework capacita o desenvolvimento de jogos pervasivos multijogador com múltiplas fontes de interação (sensores e outros aparelhos portáteis), abstraindo soluções de localização, sincronização e comunicação. Os aparelhos portáteis usados pelos jogadores são Sun SPOTs, sensores capazes de comunicação usando protocolo IEEE 802.15.4, estes que se comunicam com um nó central para monitoramento e gerenciamento do jogo. O framework FinN será analisado com mais detalhe no Capítulo 4 desse trabalho.

No caso de Hot Potato, os sensores carregados pelos jogadores são apenas capazes de enviar informações captadas pelo ambiente (pelo acelerômetro e sensor de proximidade), assim, a "inteligência" do jogo se encontra no nó central, que recebe as informações dos sensores carregados pelos jogadores. A comunicação dos sensores para o nó central é feita através de um servidor de "estação de jogo" intermediário, este capaz de mapear a distância entre ele e os

sensores de modo a ter um “mapa virtual” da posição aproximada de cada jogador.

*Can You See me Now?*

Can You See me Now (CYSMN), de Benford et al. [54], é um pioneiro jogo pervasivo multijogador baseado em localização. CYSMN foi bem impactante na área de jogos pervasivos, sendo um dos primeiros a misturar elementos virtuais e físicos no mesmo jogo. O jogo ocorre em sessões, onde 15 jogadores – os fugitivos, ou *runners* – fogem de outros 4 jogadores, os perseguidores ou *chasers*. O diferencial é que os jogadores fugitivos estão sentados em computadores, fugindo com um personagem virtual, enquanto os perseguidores estão fisicamente nas ruas, perseguindo personagens virtuais enquanto carregam aparelhos portáteis com GPS.

Os fugitivos têm acesso a um modelo virtual da cidade em seus computadores onde veem seus companheiros, os perseguidores, cuja localização é atualizada por GPS que eles carregam e podem se comunicar por mensagem de texto com seus companheiros. Já os perseguidores podem ver as localizações dos fugitivos e dos outros perseguidores em um mapa em seu aparelho portátil, além das mensagens trocadas entre os fugitivos. Os perseguidores se comunicam por walkie-talkie, e os fugitivos podem ouvir o stream do que está sendo falado entre todos os perseguidores.

A área de jogo é previamente definida como sendo um retângulo de 500 por 1000 metros. O jogo ocorre enquanto houverem perseguidores, e os fugitivos capturados recebem uma pontuação equivalente ao tempo que demorou para serem capturados desde que entraram no jogo. Apesar do tempo indeterminado, o jogo só pode ocorrer com a presença fixa de perseguidores na área de jogo, então, apesar de ser de tempo indeterminado, o jogo pode ser considerado de sessão.

A execução do jogo consiste no compartilhamento bidirecional das informações de localização e comunicação dos jogadores através dos diferentes componentes que são usados pelos jogadores (computador e aparelho portátil). O trabalho apresenta esse processo de orquestração distribuída como uma de suas contribuições, integrando a execução do jogo ao trabalho de uma equipe técnica trabalhando em uma sala separada. Os componentes usados usam conexão com a Internet, o que acarretava muitas vezes em imprecisão entre as ações dos

perseguidores e fugitivos, imprecisão esta que foi usada como vantagem por alguns jogadores, como descrito no trabalho. CYSMN, por parte da jogabilidade dos perseguidores, se enquadra no contexto de exergames móveis.

Uncle Roy All Around You

Em Uncle Roy All Around You (URAAY), de Benford et al. [52], que é uma progressão do trabalho realizado por Benford et al. [54] em CYSMN, os jogadores também são divididos entre jogadores 'na rua' e 'online'. Jogadores na rua exploram a cidade em busca de pistas para encontrar o 'Uncle Roy', enquanto jogadores online podem escolher se querem ajuda-los ou atrapalha-los, comunicando-se através de mensagens de um computador fixo.

Os jogadores móveis carregam aparelhos portáteis (PDAs) com um visor, mapa e GPS. Através dele, eles interagem com a narrativa do jogo, cumprindo objetivos e avançando à medida que se comunicam com outros jogadores e compartilham sua localização. O jogo é orquestrado através de um script fixo, onde atores humanos tomam certas ações para dar imersão aos jogadores durante a experiência. No mais, o contexto tecnológico é semelhante ou de CYSMN, onde jogadores em um ambiente indoor movimentam personagens virtuais que também interagem com os jogadores no mundo real e compartilhando mensagens de texto e vídeo entre os jogadores. A comunicação era realizada diretamente usando sistema GPRS (2G).

*Botfighters*

Botfighters, de Sotamaa [55, 57], é o único jogo dessa lista que foi lançado comercialmente. Sotamaa [55] faz uma análise dele em seu trabalho de 2002, prevendo que jogos baseados em localização multijogadores desse tipo seriam cada vez mais comuns no futuro próximo graças ao crescente uso de celulares.

Em Botfighters, jogadores criam e controlam robôs (bots) e o objetivo é localizar e destruir os robôs de outros jogadores. A jogabilidade é dividida entre computador, onde o jogador pode criar e customizar o seu robô no site oficial do jogo, e no telefone celular, para as batalhas nas ruas da cidade. O jogo em si é executado através do envio de mensagens no celular do próprio jogador, e ocorria apenas em algumas cidades específicas. Para atacar um robô, o jogador deve enviar um SMS escrito 'hunt' seguido pelo nome do jogador que deseja atacar. Os

comandos de texto para a central do jogo geram uma resposta com informações sobre a localização do robô (jogador) procurado, assim como um aviso para o jogador que ele está sendo caçado por outro. Assim, os jogadores caçados podem decidir se vão batalhar ou fugir. Quando os jogadores estão próximos o suficiente, a batalha começa com o comando 'shoot'. O jogo foi descrito na época como sendo parecido com um 'paintball virtual'.

A interação entre os usuários é realizada somente por SMSs, mediado pelo serviço telefônico que capacitava o envio das mensagens. Como o jogo era comercial, não há muitos outros detalhes a respeito da implementação do mesmo. A localização era feita por GPS tradicional.

*Human Pacman*

Human Pacman, de Cheok et al. [38], é um jogo de sessão que usa múltiplos aparelhos portáteis para dar uma experiência de jogo pervasivo imersiva e com interação física real. Nele, jogadores simulam uma versão do jogo de Pacman, onde dois jogadores são os Pacmans e outros dois são os fantasmas, juntamente com dois ajudantes para cada time. O objetivo do time dos fantasmas é 'devorar' os Pacmans, e o objetivo dos Pacmans é evitar os fantasmas enquanto procuram por 'cookies', estes que quando pegos dão o poder dos Pacmans 'devorarem' os fantasmas, o que faz com que os Pacmans vençam.

Os jogadores usam tecnologias de *wearable computing* (sensores e pequenos computadores vestíveis) com sensores de toque e Bluetooth, um laptop para interface para os sensores no corpo do jogador e o servidor de jogo e um capacete de AR. Os 'cookies' coletados pelos Pacmans são objetos físicos com um sensor Bluetooth. Esses objetos se conectam com o receptor na vestimenta do jogador, este que por sua vez avisa o servidor que muda o estado do Pacman para 'invencível' por um período de tempo. A orquestração ocorre através dos ajudantes, que carregam PDAs com mapas indicando a posição de todos os jogadores, de forma a auxiliar os jogadores de seu time a acharem ou fugirem dos do outro.

Quando um jogador fantasma toca em um Pacman, ou seja, interage com o sensor de toque na vestimenta do jogador, o Pacman é 'devorado', perdendo uma vida. Em contrapartida, se o Pacman estiver 'invencível' e tocar no sensor de um fantasma, o fantasma é que é devorado e perde uma vida. Todos os jogadores têm

duas vidas, e quando um jogador perde todas as vidas, ele sai do jogo. Quando um time fica sem jogadores, o mesmo perde. O tempo de jogo também é limitado arbitrariamente em 15 minutos.

## 3.2. Análise Comparativa

Considerando as características de contexto tecnológico e cultural descritas anteriormente, foi construída uma tabela (Tabela 3.1) comparativa dos jogos analisados nesse capítulo.

De todos os jogos analisados, o único que não se enquadra como um exergame móvel é Viking Ghost Hunt, pois nele não há suporte para múltiplos jogadores (seja colaborando ou competindo) com interação direta entre eles, nem necessidade de interação com elementos virtuais do jogo. A falta desses elementos limita os níveis de interatividade e mobilidade que é esperado de um jogador de um exergame móvel. Percebe-se também que ele é o único jogado localmente, onde toda a execução do jogo ocorre localmente, no aparelho móvel do jogador.

Percebe-se também que os jogos com interação direta entre os jogadores são justamente os jogos em que os mesmos carregam aparelhos portáteis consigo. Jogos como NetAttack, mesmo sem interação direta entre os jogadores, ainda podem ser considerados exergames móveis devido à interação indireta acarretada pela necessidade de colaboração entre diferentes jogadores de cada time para cumprir os objetivos.

A maioria dos jogos analisados são de sessão, ou seja, com pervasividade de espaço e tempo limitadas. Todos os jogos tinham pervasividade em espaço limitado, diferentemente dos jogos comerciais citados no Capítulo 1 [19, 20, 21], estes com pervasividade de espaço absoluto. Isso deve-se ao fato de serem planejados como experiências locais e não terem uma continuidade para além das interações momentâneas entre os jogadores.

Percebe-se que, no contexto tecnológico, todos os exergames móveis analisados são arquitetados de forma que o processamento das informações de jogo ocorra em um local dedicado, seja um servidor único, ou uma infraestrutura de serviços em camadas, como a implementada pelo framework FinN em Hot Potato [39].

| | Contexto Tecnológico | | | | Contexto Cultural | | | |
|---|---|---|---|---|---|---|---|---|
| | **Tecnologia empregada na execução** | **Localização** | **Visualização e Interação** | **Comunicação** | **Objetivos** | **Multijogador** | **Mobilidade** | **Pervasividade Tempo/Espaço** |
| **Time rp** | Sistema distribuído, interação c/ elementos virtuais e jogador c/ estados complexos | GPS tradicional + referência manual | MR simulado c/ capacete de AR e aparelho portátil c/ mapa | Internet/Wi-Fi | Exploração e interação c/ elementos de jogo | Não, interação c/ objetos virtuais e mudança de estado do jogador | Livre na área de jogo (cidade) e sub-regiões | Sessão de jogo: espaço e duração delimitados |
| **NetA k** | Sistema distribuído e jogadores c/ estados simples | GPS tradicional + localização implícita por marcadores AR | 1. MR simulado c/ capacete de AR; 2. Computador | Internet/Wi-Fi | Exploração | Sim, dois times (competição) de duas categorias (colaboração) | Fixa (indoor) e livre na área de jogo (delimitada) e sub-regiões | Sessão de jogo: espaço e duração delimitados |
| **Songs he No** | Sistema distribuído, interação c/ elementos virtuais e jogadores c/ estados simples | GPS tradicional | Celular c/ mapa e métodos de entrada simples | Internet/Wi-Fi ou 3G e SMSs | Exploração e interação c/ elementos do jogo | Sim (colaboração), mas não obrigatório | Livre na área de jogo (delimitada) e sub-regiões | Persiste em tempo; espaço delimitado |
| **Viking ost Hu** | Local e jogador c/ estados simples | GPS tradicional | Celular c/ mapa, MR por câmera e feedback sonoro | Internet/Wi-Fi ou 3G | Exploração | Não e mudança de estado do jogador | Livre na área de jogo (delimitada) | Persiste em tempo; espaço delimitado |
| **Hot P to** | Sistema distribuído multicamada, interação c/ elementos virtuais e jogadores c/ estados complexos | GPS tradicional + localização implícita por aproximação de sinal | Aparelho c/ tela, feedback sonoro e acelerômetro identificação de movimentos | IEEE 802.15.4 e Internet | Interação entre jogadores | Sim (competição) e interação com objetos físicos | Livre na área de jogo (delimitada) | Sessão de jogo: espaço e duração delimitados |
| **Can Y See Me ...v** | Sistema distribuído e jogadores c/ estados simples | GPS tradicional | 1. Aparelho c/ tela e MR pelo mapa; 2. Computador c/ entrada de texto e feedback sonoro | Internet/Wi-Fi e Walkie Talkie | Interação entre jogadores | Sim, duas categorias (competição) | Fixa (indoor) e livre na área de jogo (delimitada) | Sessão de jogo: espaço e duração delimitados |
| **Botfighter** | Sistema mediador de SMSs, jogadores c/ estados simples e local | GPS tradicional e por rede GSM | 1. Computador; 2. Celular c/ entrada de texto | SMSs | Interação entre jogadores | Sim (competição) | Fixa (indoor) e livre na área de jogo (cidade) | Persiste em tempo; espaço delimitado |
| **Human Pacman** | Sistema distribuído, interação com elementos virtuais e jogador c/ estados simples | GPS tradicional | MR simulado c/ capacete de AR, interação física e aparelho c/ tela e MR pelo mapa | Internet/Wi-Fi | Interação entre jogadores | Sim, dois times (competição) e duas categorias (colaboração) | Livre na área de jogo (delimitada) | Sessão de jogo: espaço e duração delimitados |
| **Uncle Roy All Around You** | Sistema distribuído e jogadores c/ estados simples | GPS tradicional | 1. Aparelho c/ tela e MR pelo mapa; 2. Computador c/ entrada de texto e visualização por vídeo | Internet/2G | Exploração, interação c/ elementos de jogo | Sim (colaboração e competição) | Fixa (indoor) e livre na área de jogo (delimitada) | Sessão de jogo: espaço e duração delimitados |

Tabela 3.1: Análise das características dos exemplos de jogos no contexto tecnológico e cultural

# 4
# Middleware para Exergames Móveis

Esse capítulo apresenta o conceito e implementação de um middleware que capacite a execução de exergames móveis dadas as suas características e requisitos de execução e orquestração, analisados nos capítulos anteriores. A orquestração de jogos pervasivos consiste na operação e gerenciamento do jogo 'ao vivo', como descreve Benford et al. [11]. A arquitetura do middeware deve ser capaz de gerenciar sessões de diferentes jogos, os jogadores participantes de cada uma das sessões e seus estados no jogo e os objetos físicos existentes. Além disso, tratando-se de exergames móveis, ela deve também prever a implementação de serviços diferentes para resolver localização com estratégias e/ou tecnologias distintas e regras para diferentes tipos de jogos, semelhante à arquitetura em camadas proposta no trabalho de Akribopoulos [39].

O middleware proposto resolve parcialmente algumas questões de interoperabilidade [58] entre as diversas tecnologias envolvidas no desenvolvimento desses jogos, separando as responsabilidades de interface com os elementos do jogo em diferentes camadas. Apesar disso, o middleware apresentado tem como objetivo maior possibilitar o estudo de caso de execução da API de microlocalização proposta não sendo, assim, o real foco deste trabalho.



## 4.1.
## Trabalhos Relacionados

Nessa seção, são analisados alguns trabalhos que têm em sua proposta soluções relacionadas a execução de jogos pervasivos. Alguns deles preveem desafios únicos de exergames moveis, já outros são menos compatíveis.

Freegaming

O trabalho de Görgü et al. [18] inicialmente define Freegaming como um conceito, uma categoria de jogos baseada em exergames com mobilidade, uso de tecnologias AR, multijogador e adaptatividade em ambientes outdoor. Nele, uma plataforma para o desenvolvimento de jogos desse tipo (também chamada de

Freegaming) é proposta. A plataforma é implementada baseada em um framework de código aberto chamado ‘Agent Factory Mobile Edition’ (AFME [62]).

A plataforma se divide em dois componentes, o Servidor de Freegaming e a Aplicação Móvel de Freegaming (FMA) em um contexto de sistemas distribuídos. O servidor de Freegaming consiste em dois servidores físicos, o de jogo (FGS) e o de visão de máquina (MVS). O FGS lida com a execução do jogo e o gerenciamento dos jogadores através de um banco de dados próprio para estas informações. O MVS lida com os aspectos de AR dos jogos como gerenciamento das tags e vínculo entre tag e componente virtual a ser mostrado aos jogadores. Já o FMA lida com os inputs do usuário, posicionamento por GPS e atualiza os estados do jogo e do jogador com base nas informações de ambiente que pode capturar (caso tenha outros tipos de sensores). Ele também é responsável por apresentar os objetos AR na tela de jogo através do uso de uma câmera interna.

Em Freegaming, os servidores e clientes de jogo são capazes de executar uma instância de um tipo de jogo por vez, havendo necessidade de uma arquitetura de múltiplos servidores caso diferentes jogos com múltiplas sessões sejam executados ao mesmo tempo. Além disso, as características de uso de tecnologias são fixas (AR, GPS, etc), não havendo muito espaço para uso de diferentes tecnologias ou técnicas para solucionar problemas de localização e interação de entre jogadores, por exemplo.



FinN

FinN, de Akribopoulos et al. [39], é um framework para desenvolver aplicações pervasivas e instalações interativas para propósito educacionais e de entretenimento. O framework é planejado em camadas, sendo elas a camada “guardiã”, que é composta pelos aparelhos móveis carregados pelos jogadores, estes podendo ter interfaces e métodos de entrada variáveis; a camada de “estação de jogo”, onde a infraestrutura dos serviços necessários para a execução do jogo se encontra, como resolução de localização e ciência de contexto para os jogadores; a camada de “motor de jogo”, onde ocorre a instanciação do sessão de jogo e a coordenação das regras e condições de vitória do jogo, além do gerenciamento dos dados dos jogadores, ambiente e elementos dos jogo; e a camada de “mundo”, que gerencia os diferentes tipos de jogos FinN, sites e usuários. Esta arquitetura pode ser vista na Figura 4.1.

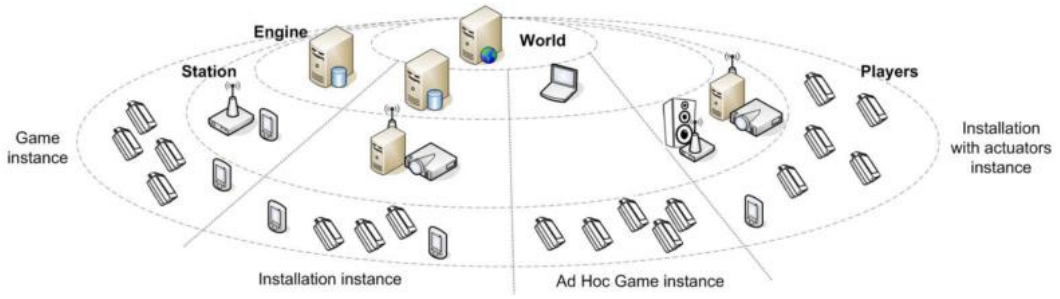


Figura 4.1: Arquitetura geral do framework FinN, como apresentada em [39]

O framework possibilita a execução do jogo em múltiplas camadas, cada uma responsável por diferentes aspectos do jogo. Exemplos são a resolução de localização ou processamento das condições de vitória, dados dos jogadores e gerenciamento de sessões de jogo. Este fato possibilita uma maior interoperabilidade entre os diversos componentes que compõem a arquitetura, já que diferentes soluções tecnológicas podem ser implementadas e usadas em diferentes jogos, dependendo das necessidades ou contexto dos mesmos.

FinN é um excelente exemplo de proposta de framework para exergames móveis e o trabalho aqui apresentado foi largamente inspirado nele. A diferença deste trabalho se dá através do uso de uma plataforma comum, o Smartphone, o foco na execução do jogo através de sessões e o foco tecnológico em resolver questões referentes a localização dos jogadores.

FRAP

FRAP, de Tutzschke et al. [61], se apresenta como um framework específico de domínio para jogos pervasivos, focado em jogos cientes de contexto, multijogadores e de ‘perseguição’. É implementado para Smartphones executando Android. A execução do jogo proposta pelo framework inclui definição de times de jogadores, da área de jogo e de objetos físicos que possam ser usados no jogo para interação com os jogadores. O servidor de jogo é capaz de guardar informações dos jogadores (inclusive localização, essa enviada do Smartphones dos jogadores por GPS), tratar as regras do jogo e criar canais de comunicação entre os jogadores. A arquitetura proposta segue o modelo de sistemas distribuídos e a comunicação entre clientes (jogadores) e servidor é feita diretamente (HTTP) pela Internet.

Morgan

Usado por ambos NetAttack [53] e TimeWarp [49], Morgan, de Ohlenburg et al. [59], é apresentado como um framework para projetos de AR e VR. O trabalho propõe resolver desafios de escalabilidade, interoperabilidade, gerenciamento de múltiplos usuários e distribuição de componentes para aplicações que usam tecnologias de AR e VR através de um framework baseado em componentes. O framework inclui uma API com opções de componentes de input para os jogadores, aparelhos de rastreamento para AR e motor gráfico modular para jogos em que se façam necessários.

O framework é específico para aplicações que usam AR/VR, e de fato dispõe de opções de integração para vários aparelhos e softwares terceiros dentro do escopo dessas tecnologias, porém não propõe uma arquitetura que leve em consideração outros tipos de tecnologias, nem leva em consideração questões chave execução de jogos que consideramos importantes neste trabalho (sessões de jogo e gerenciamento de jogadores por exemplo), dada a sua natureza genérica para aplicações AR/VR, e não necessariamente jogos.



## 4.2. Responsabilidades do Middleware

Como anteriormente comentado, um middeware para exergames móveis deve idealmente ter uma arquitetura capaz de gerenciar diferentes componentes físicos e virtuais e sua interoperabilidade de forma que o jogo possa ser iniciado, executado (seguindo as regras do mesmo) e finalizado (dada condições de término) da maneira esperada.

Tipicamente, o middleware deve definir uma arquitetura de sistemas distribuídos contendo componentes responsáveis por:

- Criar, modificar e monitorar *sessões de jogo*, com vínculo para o tipo de jogo, com definição de área de jogo, número de jogadores, times e sub-regiões internas a área de jogo;
- Dar capacidade de manipular e monitorar os *elementos virtuais* do jogo, suas informações e estados, sejam eles os jogadores, objetos virtuais, personagens não jogadores e eventos ‘scriptados’;

- Gerenciar e monitorar os *elementos físicos* do jogo, sejam eles jogadores (em que sessão de jogo estão conectados, estatísticas, etc.), objetos físicos (tags, sensores, beacons, marcadores AR) e seus papéis (mapear sub-regiões para localização, interação com jogadores, etc.);
- Gerenciar as *relações entre os elementos do jogo*, os tipos de interações possíveis e as regras de interação entre jogadores e objetos físicos;
- Declarar e vincular os componentes específicos para cada tipo de jogo que resolvem certos aspectos do mesmo, podendo ser:
    - Componente *executor das regras* para um determinado tipo de jogo;
    - Componentes específicos para as soluções de tecnologia necessárias para cada tipo de jogo: *Localização*, *comunicação entre jogadores* e *AR*, podendo haver outros;
    - Adicionalmente, cada componente tecnológico que requeira um módulo ou serviço de resolução própria deve definir uma API comum de interação com o módulo central que executa o jogo, de forma que o modulo executor possa ter chamadas comuns de API independente da tecnologia que está capacitando o jogo. Uma das propostas desse trabalho é a API para localização, descrita no Capítulo 6.



### 4.2.1. Sessões de Jogo

Uma sessão de jogo exergame móvel segue a definição do Capítulo 2, ou seja, deve ter um local definido, um momento de início e um fim. Para tal, foram definidos alguns estados possíveis para a sessão de jogo, mostrados na Figura 4.2.

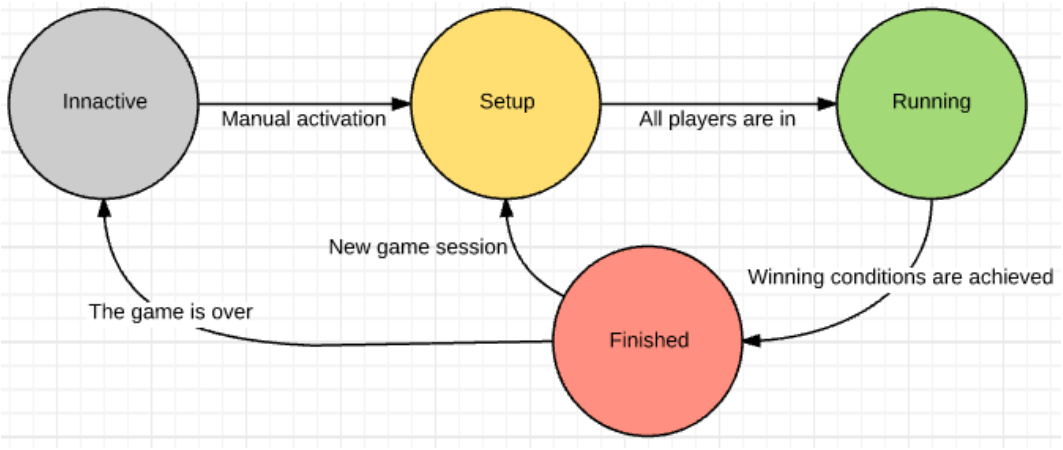


Figura 4.2: Estados possíveis da sessão de jogo

- *Desativada*: A sessão foi criada, mas ainda não pode ser acessada por jogadores. A sessão pode ir de ‘desativada’ para ‘em configuração’ manualmente pelo administrador do jogo.

- *Em configuração*: A sessão foi criada e pode ser acessada por jogadores até o limite da capacidade. A sessão de jogo é automaticamente iniciada ao término da fase de configuração, que por padrão é quando todos os jogadores entraram no jogo.

- *Em andamento*: O jogo está em andamento e ficará assim até o seu término. Após essa fase, o jogo vai para a fase de ‘terminado’.

- *Terminado*: Condições de vitória ou derrota foram alcançadas pelos jogadores, o que causou o término do jogo. Após isso, o jogo pode voltar para a fase de ‘em configuração’ para o início de uma nova sessão ou ser manualmente ‘desativada’.

Sessões de jogo são criadas no serviço de administração do middleware, este manualmente controlado por um administrador. Nele, dado os tipos de jogos disponíveis, o administrador gerencia e monitora sessões de diferentes tipos de jogos. Uma sessão de jogo pode ser definida pelo seguinte modelo de dados, descrito no padrão JSON na Figura 4.3.

```
game_session = {
  'default_state': 'innactive',
  'game_type': {
    'game_id': INTEGER,
    'game_name': STRING,
    'game_tag': STRING
  },
  'teams': LIST,
  'game_area': {
    'coordinates': LIST,
    'radius': INTEGER,
    'sub_regions': LIST
  }
}

teams = [{
  'teams_id': INTEGER,
  'team_name': STRING,
  'team_num_players': INTEGER
}]

sub_regions = [{
  'coordinates': LIST,
  'radius': INTEGER,
  'beacon_mapping': LIST
}]
```

Figura 4.3: Dados relativos a uma sessão de jogo

Os times são definidos pela quantidade de jogadores e por um identificador. Como o middleware é pensado de forma que o administrador possa executar sessões de diferentes tipos de jogo, o valor ‘game_type’ contém as informações necessárias para que se saiba qual o tipo de jogo será instanciado. Através dessa identificação, o middleware sabe o serviço de regras específico do tipo de jogo. As definições de área de jogo e sub-regiões serão discutidas nas próximas sessões.

### 4.2.2. Área de Jogo

A área de jogo é definida no momento da criação da sessão. Um exemplo de área de jogo pode ser visto na Figura 2.1. A área é delimitada por um raio em metros, sendo circular e centrada em uma coordenada latitude/longitude. O jogo ocorre dentro da área de jogo, porém, o desenvolvedor do exergame móvel é livre para invalidar ações do jogador fora da área de jogo ou incluir isso na jogabilidade e/ou regras de jogo da forma que desejar. A forma com que o jogador é localizado dentro da área de jogo vai variar dependendo da tecnologia de localização prevista

para uso no tipo de jogo, porém a API de localização prevê a localização do jogador na área de jogo por coordenada latitude e longitude.

### 4.2.3.
### Mapeamento das sub-regiões de microlocalização

Sub-regiões de jogo podem ser definidas dentro da área de jogo. Tipicamente, uma sub-região é circular e centrada em uma coordenada geográfica, como uma região de jogo, porém, contida na área (sub-regiões necessariamente devem ser menores que a área de jogo e contidas na mesma). Opcionalmente, uma sub-região ser vinculada a uma lista de objetos físicos, como beacons, estes que podem ser usados pelo componente de localização do jogo para resolver a microlocalização do jogador dentro dessa região.

### 4.2.4.
### Relacionamento entre elementos de jogo

O gerenciamento e mapeamento dos objetos físicos e virtuais e suas interações pode ser realizado da maneira que o desenvolvedor do jogo desejar. Todavia, o middleware prevê que jogadores e objetos físicos usados em um tipo de exergame móvel necessariamente tenham papeis definidos, podendo ser, por exemplo, o time de um jogador, e que há interações entre papeis de elementos diferentes do jogo, como descrito no modelo da Figura 4.4.

```
relation_mapping = {
  'objects': {
    'obj_type_1': LIST,
    ...
  },
  'players': {
    'player_type_1': STRING,
    ...
  },
  'relations': LIST
}

relations = [{
    'player_type_1': {
      'with': 'obj_type_1',
      'interaction': STRING
    }
    ...
}]
```

Figura 4.4: Dados relativos as relações entre elementos do jogo – objetos físicos e jogadores. O tipo de objeto é mapeado diretamente para uma lista de identificadores de objetos, enquanto que o tipo de jogador é mapeado para um 'team_name' definido na sessão de jogo.



### 4.2.5.
### Instância de jogo e 'resolvedor' de regras

A instância de jogo é criada juntamente com a sessão, e nela deve haver um serviço que lide com as regras e condições de vitória do jogo. Esse serviço pode ser local, executando em um 'thread' separada para cada instância do tipo de jogo, por exemplo, ou em um servidor separado, dependendo da infraestrutura desejada. É neste serviço que as informações e estados dos jogadores, objetos do jogo, eventos, NPCs e outros elementos do jogo são guardados e atualizados, lidando com o conjunto de dados necessários (banco de dados, tabelas, arquivos ou outros componentes de dados dependendo da implementação desejada). Considerando todas as informações de jogo e regras do mesmo, este serviço deve ser capaz de atualizar o estado do jogo e determinar o seu término dada condições de vitória.

### 4.2.6.
### Camada de comunicação e APIs para tecnologias usadas

Os dados recebidos dos jogadores, que estarão tipicamente usando Smartphones ou outros aparelhos portáteis, são recebidos em um servidor de administração de sessões de jogo no qual eles se conectam diretamente. Dependendo da infraestrutura desejada, a responsabilidade de conectar os

jogadores aos serviços de tecnologia e às instâncias de sessões de jogos poderia ser realizada de maneira distribuída entre múltiplos servidores, de forma a tornar a execução de múltiplas sessões de múltiplos tipos de jogos escalável. As APIs para as soluções tecnológicas, que variam dependendo do tipo de jogo e de sua implementação, idealmente devem ser pré-definidas para que as implementações das soluções tecnológicas se adequem para dialogar com elas. Nesse trabalho, no Capítulo 6, descreveremos uma proposta para uma API de localização que considera microlocalizações em sub-regiões. APIs para outros tipos de soluções tecnológicas, como AR, não serão tratadas nesse trabalho.

## 4.3. Arquitetura

A arquitetura proposta para o middleware de exergames móveis se assemelha a uma simplificação da arquitetura em camadas proposta por Akribopoulos et al. [39] com o framework FinN. O encapsulamento de APIs em serviços especializados (no caso, somente a de localização foi planejada, a ser descrita no Capítulo 6) é inspirado no padrão de *Service Oriented Architecture* (SOA), apesar da troca de dados entre os serviços ser através de 'stream' de dados, usando o middleware SDDL, e não SOAP ou REST, protocolos mais comuns a serviços web.

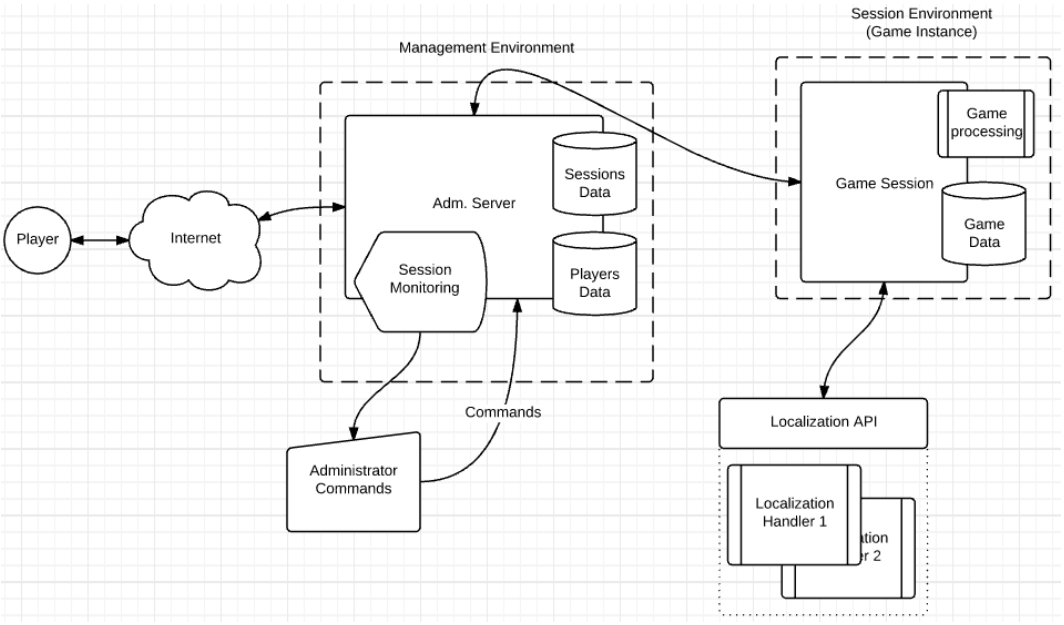


Figura 4.5: Arquitetura do middleware proposto

Na arquitetura proposta, os aparelhos móveis dos jogadores enviam informações de jogo, do ambiente e do jogador e suas ações (automaticamente ou por entrada manual do jogador). Essas informações são transmitidas pela Internet para um servidor de administração, que pode ser distribuído em uma estrutura de múltiplos servidores caso escalabilidade seja necessária. O servidor de administração tem as informações de conexão do jogador, que inclui qual jogo ele está jogando, sabendo assim retornar para o jogador quais sessões desse jogo estão disponíveis para ele. Ao mesmo tempo, um ou mais administradores do ambiente têm acesso a esses servidores, podendo criar novas sessões de diferentes tipos de jogos e monitorar conexão, localidade e estado em jogos dos jogadores, além de monitorar o estado de sessões de jogos, registrar elementos físicos e configurar futuras sessões.

As instâncias de sessões de jogos podem ocorrer localmente no próprio servidor de administração ou serem remotamente iniciadas em outro servidor dependendo da infraestrutura desejada. O servidor administrativo tem a informação de qual sessão de jogo o jogador está em, sabendo assim redirecionar as informações recebidas do jogador acerca do estado do jogo para a instância de jogo. As sessões de jogo recebem as informações atualizadas dos estados dos jogadores, objetos físicos, ambientes e eventos do jogo, lidando com elas de formas diferentes dependendo da implementação do tipo de jogo (suas regras e condições de término). Quando necessário, requisições de APIs especificas de tecnologias usadas no tipo de jogo em questão são feitas diretamente da instância de jogo. A API resolverá a requisição usando a estratégia correta para o tipo de jogo da instância.



## 4.4. Implementação

Foi realizada uma implementação do middleware através de um servidor escrito na linguagem Java. Em conjunto, foi desenvolvido um aplicativo Android capaz de se conectar ao servidor, enviar informações do jogador e do aparelho, apresentar informações recebidas do servidor na tela, entrar e sair de uma sessão de jogo e fazer requisições da API de localização.

O servidor de administração é acessado por um administrador, que pode atuar como monitorador ou como orquestrador do jogo através de uma interface de linha de comando (Figura 4.6). Através dele, o administrador pode criar, configurar e ativar sessões de jogo genéricas, listar sessões criadas, verificar jogadores conectados e se eles estão em alguma sessão de jogo (e seu time no jogo), gerenciar objetos físicos (beacons BLE na implementação realizada) e vinculá-los à sub-regiões em diferentes sessões de jogo. O administrador também pode mandar mensagens para os jogadores manualmente.

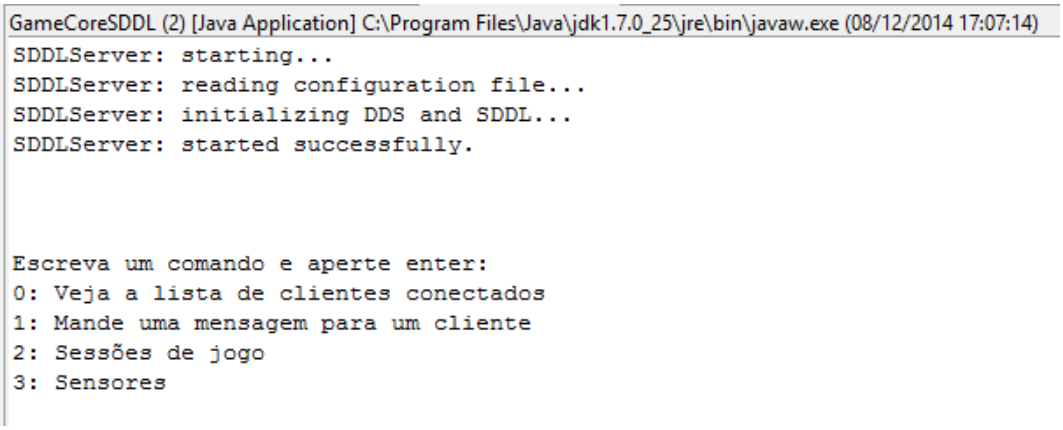


Figura 4.6: Interface de linha de comando do administrador de jogo

O restante das classes e módulos do ambiente implementado segue a arquitetura proposta na Figura 4.5, como pode ser visto no diagrama de classes UML apresentado na Figura 4.7. As sessões de jogo são vinculadas a ‘threads’ separadas sendo executadas no mesmo servidor administrativo e gerenciadas por um ‘SessionThreadManager’. Como o acesso do ‘AdminServer’ é realizado diretamente através do gerenciador de threads, o acesso a API de localização, que implementa a interface ‘ILocationResolver’ é realizado diretamente também, ao invés de através da sessão de jogo como a arquitetura propõe. Alguns componentes, como ‘CmdLineInterpreter’ (responsável pelo método de entrada de dados e monitoramento do administrador), ‘GameDDSLayer’ (responsável pela conexão e troca de mensagens), ‘GameElementRole’ (modelo de dados que define o papel do jogador na instância de jogo) e ‘GameHandler’ (responsável por processar os dados e estados dos jogadores e do jogo e mudar os estados da sessão de jogo com base nas regras e critérios e término) foram omitidos do diagrama

pois ou não tinha relevância para a análise ou pois são específicos da implementação do tipo de jogo desejada.

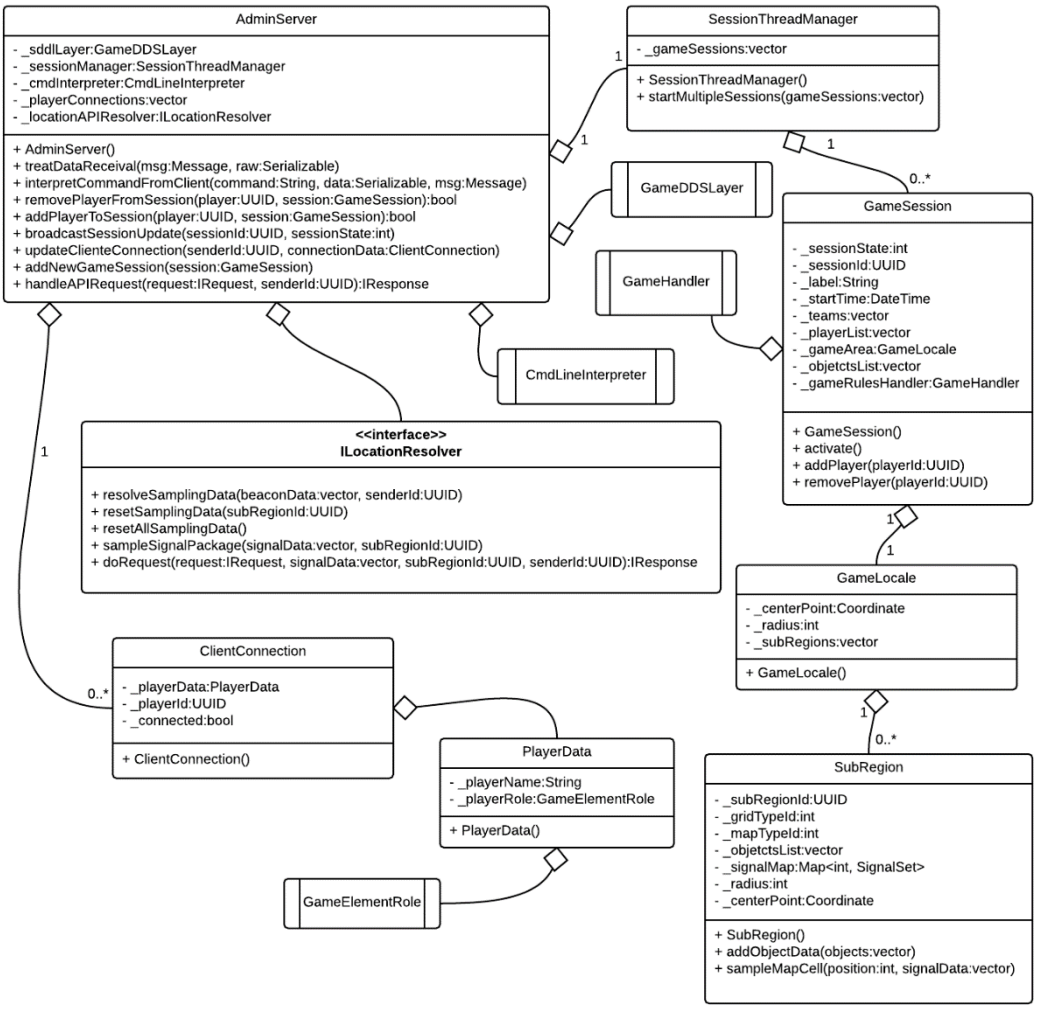


Figura 4.7: Diagrama de classes UML da implementação realizada do middleware

### 4.4.1.

### M-Hub e o SDDL

O módulo ‘GameDDSLayer’ é responsável pela comunicação entre os clientes móveis (Smartphones carregados pelos jogadores) e o servidor de administração. Essa comunicação é realizada através de um middleware de comunicação baseado em ‘Publish/Subscribe’ proposto por David et al. [63] chamado *Scalable Data Distribution Layer*, ou *SDDL*.

O middleware SDDL se enquadra um projeto maior, apresentado por Endler et al. [64] como *ContextNet*. O projeto ContextNet é focado primariamente em desenvolver serviços de middleware e APIs de forma a facilitar o

desenvolvimento em larga escala de aplicações pervasivas cientes de contexto. O middleware SDDL é um dos projetos inclusos no ContextNet, que se encontra em constante evolução[2].

O middleware SDDL é capaz de conectar nós estáticos, como o servidor de administração, a nós móveis, como os Smartphones carregados pelos jogadores. O protocolo de comunicação usado pelo SDDL é o Mobile Reliable UDP, ou MR-UDP, proposto por Silva et al. [65]. O protocolo tem mecanismos capazes de lidar com conectividade intermitente, mudanças em endereços de IP dinâmicos dos nós móveis e alcance a nós através de firewalls e NATs. Este protocolo é usado pelos nós móveis para conexão com um nó estático de *Gateway*, podendo haver vários gateways com diferentes IPs para uma mesma aplicação distribuída. Cada gateway mantém uma conexão MR-UDP independente para cada nó estático interno da arquitetura do SDDL, como pode ser visto na Figura 4.8. Além disso, os gateways são responsáveis por garantir a entrega das mensagens entre nós móveis e nó estáticos.



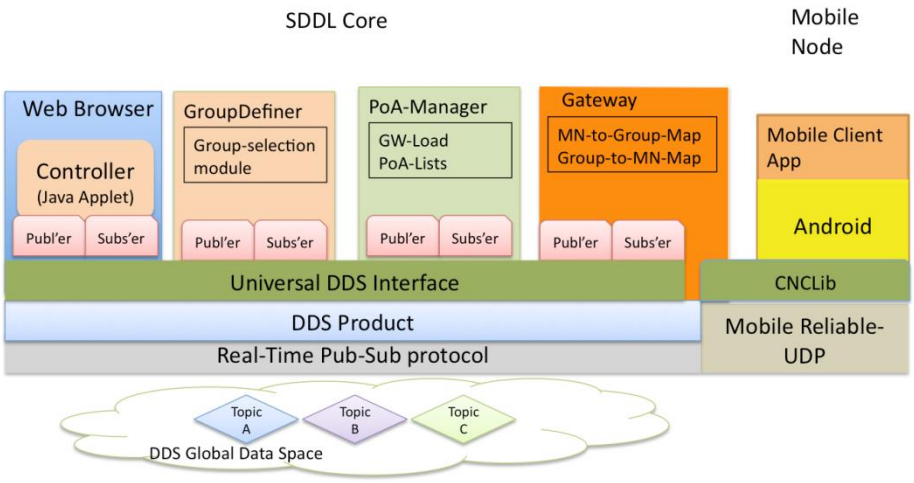


Figura 4.8: Nós estáticos e camadas do SDDL, como visto em [63]

Como pode ser visto na Figura 4.8, o SDDL contém outros tipos de nós estáticos especializados que não serão descritos nesse trabalho pois não são usados no ambiente proposto (mais informações em [63]).

A comunicação da parte do cliente móvel é realizada através de uma biblioteca especializada chamada *ClientLib* (ou *CNClib*). A biblioteca é escrita em

[2] Informações de projetos e downloads em http://www.lac-rio.com/projects e http://www.lac-rio.com/software

Java e lida com conexões MR-UDP de clientes móveis e atualmente compatível com aplicações nativas Java, na plataforma Android e Lua. O uso da biblioteca no aplicativo de testes para Smartphones Android foi realizado internamente através do uso da biblioteca Mobile-Hub, ou M-Hub, apresentada por Talavera et al. [66]. O M-Hub não somente usa a ClientLib internamente para comunicação com o servidor, mas também implementa serviços de localização para uso interno do aplicativo móvel, gerenciamento inteligente de energia do aparelho e serviços para identificação e conexão com objetos físicos de tecnologia WPAN. É através desses serviços do M-Hub que os Smartphones são capazes de se comunicar com objetos físicos, o que por consequência possibilita a inclusão desses objetos em experiências de jogo e/ou para mapear sub-regiões de jogo para localização implícita. Um sumário da arquitetura do M-Hub pode ser visto na Figura 4.9.



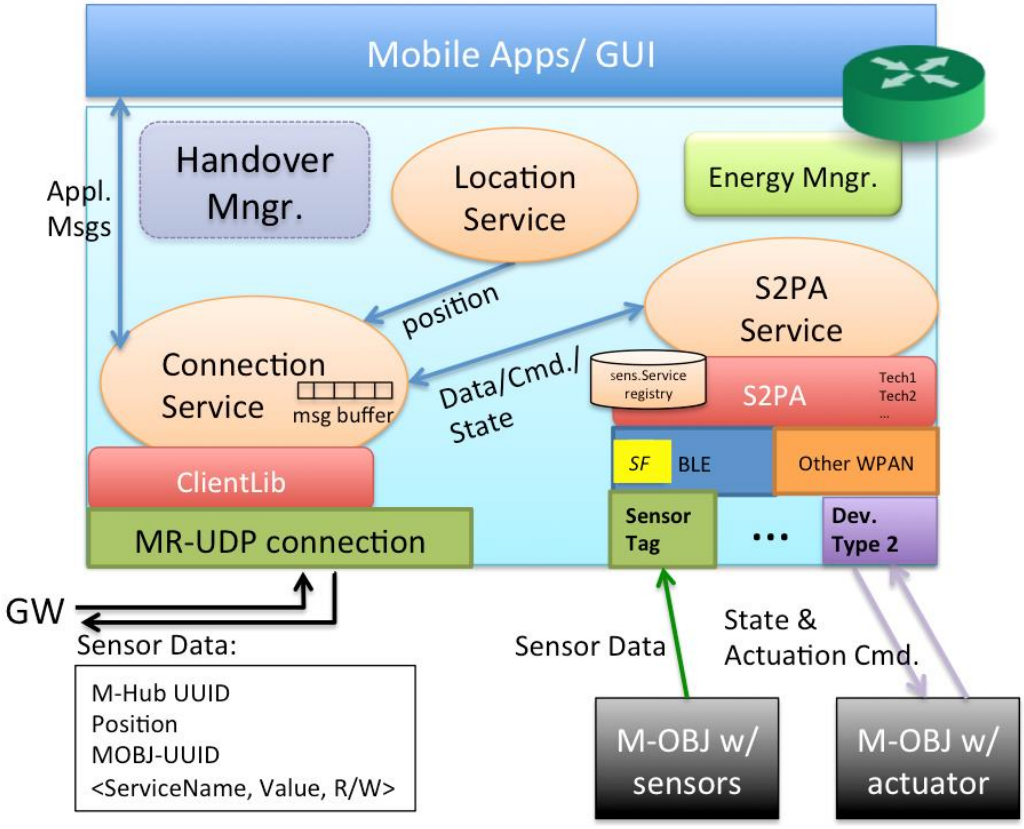


Figura 4.9: Arquitetura do Mobile Hub

Tanto o SDDL quanto o M-Hub foram testados em experimentos controlados em laboratório no que diz respeito a confiabilidade, escalabilidade e desempenho [63, 66], tendo resultados bem satisfatórios para o escopo do

middleware de jogos. Na análise feita por David et al. [63], usando atualizações de localização de um Smartphone Android, conseguiram um tempo médio de troca de mensagens entre 20 e 45ms com o sistema sob testes estresse. Esse tempo indica que, com o uso do Middleware do SDDL, o gargalo da inferência de localização por parte do envio de dados do Smartphone certamente não ocorreria no envio dos dados.

## 4.5. Discussão

Este capítulo apresentou uma proposta de arquitetura para um middleware de exergames móveis. Esta proposta engloba a resolução de alguns problemas de interoperabilidade, tangenciando responsabilidades de um middleware, apesar disto não ser tomado como parte da real contribuição. Em conjunto com a proposta de arquitetura, foi apresentada uma implementação usando certas tecnologias como um servidor escrito em Java, cliente móveis em Smartphones da plataforma Android e uso do middleware SDDL para comunicação e do Mobile-Hub para integração com objetos físicos capacitados por WPAN.

O ambiente proposto serve seu propósito de atuar como capacitador para os testes da estratégia de localização implícita através do uso da API de requisições e eventos, temas dos Capítulos 5 e 6. Apesar de influenciado pelo trabalho de Akribopoulos et al. [39] com o framework FinN, o ambiente não se propõe a ser uma solução melhor que a proposta nesse trabalho, e sim ser apenas uma reinterpretação do que foi proposto em um contexto ligeiramente diferente.

Dito isso, há alguns pontos que poderiam ser trabalhados em possíveis melhorias futuras, tanto na arquitetura proposta como em uma implementação que a segue com maior fidelidade. O módulo de ‘GameHandler’, apresentado na arquitetura, poderia ser melhor definido, de forma a capacitar a inclusão dinâmica de regras e condições de vitória, além de integrar melhor as relações de interação entre jogadores e objetos físicos no jogo através do modelo ‘GameElementRole’ ou de algum módulo interno responsável por estas interações.

Também seria interessante definir com maior clareza uma arquitetura em camadas semelhante à usada em FinN, de modo a contribuir para escalabilidade, distribuição de carga e interoperabilidade. Através disso, seria possível definir interfaces para uso de diferentes tecnologias ou middlewares de comunicação,

além da de localização. Adicionalmente, uma exploração do uso de tecnologias AR e uma subsequente definição de uma API para diferentes implementações de soluções nessa área, nos moldes da de localização, seria um avanço interessante para capacitar o desenvolvimento de exergames móveis de realidade mista.

Da parte de implementação, a mesma atualmente não segue à risca o que foi proposto na arquitetura. Seria oportuno o desenvolvimento de diferentes implementações com múltiplas soluções de comunicação, localização e AR para diferentes tipos de jogos como forma de validar e consolidar a arquitetura proposta para o desenvolvimento de exergame móveis.

# 5
# Fingerprinting usando Beacons BLE

Neste capítulo, apresentamos o conceito por trás de localização implícita, analisando trabalhos relevantes na área de localização em tempo real, sendo alguns deles dentro do paradigma de sistemas de redes de sensores sem fio. Em seguida, analisamos o conceito e implementação de um sistema de localização implícita outdoor baseado em fingerprinting por amostragem com beacons BLE. Essa técnica é apresentada como prova de conceito para a API de requisições e eventos descrita no Capítulo 6. Finalmente, são analisados testes de campo feitos usando tecnologias específicas e seus resultados são discutidos.

## 5.1.
## Trabalhos Relacionados

Nessa seção, são analisados trabalhos que propõem estratégias de localização implícita para serviços baseados em localização em que o GPS convencional não resolve o problema. Para isso, foram pesquisados sistemas de localização relacionados ao uso de sensores, beacons e radiofrequência, tanto para adquirir uma localização precisa quanto aproximada.

O trabalho de Bulusu et al. [67] propõe uma solução de localização baseada em beacons com tecnologia de radiofrequência (RF) ‘ad-hoc’, aproveitando a existência prévia de beacons em um determinado local outdoor percorrido. Sua proposta mantém a responsabilidade de determinar a localização no próprio receptor dos sinais de RF, além de manter a exigência de baixa latência e uso de energia, dado que os beacons usados têm recursos limitados. O trabalho categoriza a estratégia proposta como de localização grosseira *(coarse-grained localization)*, onde o resultado encontrado é o próprio ponto de referência mais próximo, diferente das estratégias de localização refinada (*fine-grained localization*), em que o resultado é uma distância aproximada de um ponto de referência mais próximo, como mostrado na Figura 5.1.

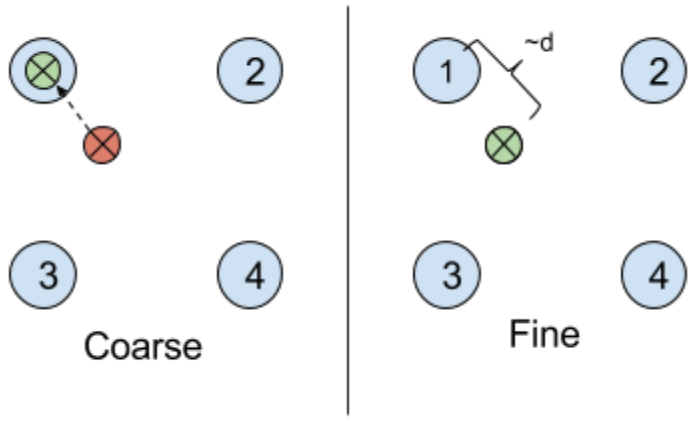


Figura 5.1: Localização grosseira e refinada. Na grosseira, a localização aproximada da bolinha vermelha é o ponto de referência, enquanto que na fina é um ponto calculado a uma certa distância do ponto de referência (sobrepostas na imagem)

O algoritmo de localização usado considera as posições de referência fixas dos beacons em uma geometria de malha regular. Ao capturar os sinais dos beacons e sabendo a posição dos pontos de referência, o nó móvel é capaz de se localizar automaticamente na malha dado prévio conhecimento das intensidades médias de sinal e alcance de cada beacon. Essa estratégia não requer amostragem prévia, porém requer conhecimento prévio das especificações técnicas dos beacons usados e dos sinais transmitidos. O autor cita que a estratégia escolhida é melhor executada outdoor dada a menor ocorrência de interferências causadas por reflexões e oclusões.

O trabalho de Chu et al. [68] assemelha-se ao de Bulusu et al. [67], também propondo uma estratégia de auto localização por parte do receptor dos sinais dos sensores, porém através de um modelo de sobreposição de células ao invés de aproximação de sinal de rádio. Sabendo previamente os raios de ação de cada beacon e suas posições, é possível determinar em qual célula ou subseção da zona mapeada o receptor móvel se encontra, dado que essas células consistem na interseção de raios de ação de múltiplos beacons.

O trabalho de Boukerche et al. [69] aponta diferentes componentes a serem levadas em consideração para algoritmos de sistemas de localização em redes de sensores sem fios, como estimação do ângulo e distância entre receptor e beacon, intensidade do sinal recebido (RSSI), diferença do tempo de chegada e ângulo de chegado. Cada um desses componentes é usado em diferentes soluções para diferentes contextos. Quando é usado o método de trilateração, por exemplo,

receptores usam informações das distâncias aproximadas de outros três nós de referência para estimar sua própria, enquanto que no método de triangulação, os ângulos relativos entre o receptor e três nós de referência é que são usados. Boukerche et al. [69] também comenta sobre o método de *fingerprinting*, onde as informações de sinais dos nós de referência são catalogadas por amostragem prévia para a construção de uma tabela. Assim, o posicionamento é realizado através da comparação entre os sinais recebidos e as amostras prévias da tabela.

O método de fingerprinting é mais comumente explorado em trabalhos de localização indoor e muitas vezes fora do contexto de redes de sensores sem fios, pois costuma ser realizado usando pontos de acesso Wi-Fi ao invés de beacons de radiofrequência. O problema citado como maior impeditivo para a sua execução no contexto de localização indoor são as interferências 'multipath', onde os sinais chegam nos receptores por múltiplos caminhos devido a oclusão e reflexão dos mesmos, como citado em diversos trabalhos [70, 71, 72, 73, 74, 75].

Para este trabalho, escolhemos usar beacons BLE para analisar o desempenho de um sistema de localização outdoor baseado na estratégia de fingerprinting. Como mostrado no trabalho de Faragher [72], é possível obter bons resultados com o método de fingerprinting usando beacons BLE em localização indoor, onde há mais problemas de oclusão e reflexão de sinal do que em um sistema executado outdoor. Usamos as conclusões atingidas por Faragher em alguns pontos da nossa solução, como pode ser visto nas seções seguintes.



## 5.2.
## Fingerprinting por amostragem usando BLE

Os trabalhos descritos na seção anterior em sua maioria descrevem técnicas e estratégias derivadas de três métodos principais de localização implícita: triangulação, trilateração e fingerprinting. Triangulação e trilateração são métodos que derivam a localização de um usuário a partir do ângulo ou distância aproximada deste para beacons ou pontos de acesso Wi-Fi com posições fixas conhecidas que mapeiam a região de interesse. Estes métodos são implementáveis dado que se conheça previamente as relações entre as informações capturadas em tempo real (intensidade de sinal recebido e transmitido, ângulo de recepção e transmissão e diferença do tempo de chegada

são exemplos) e sua correlação com informações úteis para posicionamento, como distância e ângulo aproximado.

Fingerprinting, por outro lado, inclui uma fase prévia de amostragem em que as intensidades de sinal em posições fixas da região mapeada por beacons são guardadas em uma tabela para referência futura. Assim, a única informação prévia necessária durante a execução da estratégia é a própria tabela, de modo a compararmos os resultados obtidos a cada momento com os previamente coletados para fazermos uma aproximação da posição do usuário.

Problemas como 'multipath', variância do RSSI devido a dependência da potência do sinal transmitido do beacon ao fornecedor e carga de bateria do mesmo e interferências de sinais externos dependentes do ambiente, como Wi-Fi e outras ondas de rádio, tornam pouco confiáveis relações diretas entre RSSI e distância entre transmissor e receptor, necessária para os métodos de trilateração e triangulação.

Como já citado, a técnica de fingerprinting é mais comumente usada para soluções de localização indoor usando pontos de acesso Wi-Fi [70, 71, 73, 74, 75]. A relação dessa técnica com soluções indoor deve-se principalmente a dificuldade de preparo da infraestrutura relativa a demarcação da área monitorada, o que dificulta a escalabilidade para áreas outdoor. Apesar desta limitação, a escolha de usarmos essa estratégia de localização não vai contra o objetivo deste trabalho, já que estamos interessados em mapear apenas pequenas áreas de um jogo outdoor usando beacons de WPAN (especificamente BLE) ao invés de pontos de acesso Wi-Fi.

Sun et al. [35] argumentam que, apesar de fingerprinting ser recomendada para superar problemas relacionados a multipath, o seu uso em ambientes outdoor é problemático dada a natureza menos estável e segura do ambiente de rádio, o que atrapalha a fase off-line. Martin [74] comenta sobre as vantagens de se poder trabalhar com diferentes modelos e algoritmos probabilísticos dependendo da precisão desejada quando lidando com a comparação das amostras em fingerprinting.

Tratando-se de BLE, Stojanović [33] cita BLE como sendo uma alternativa de alta precisão (< 0.3m dependendo da estratégia de localização, topologia e fatores externos) e baixo custo, apesar da necessidade de infraestrutura preparada (demarcação da área) e necessidade de visibilidade dos beacons ('line

of sight'). Por outro lado, Faragher [72] argumenta que, pelo fato de BLE transmitir em 3 canais de frequência diferentes, os RSSIs sequenciais muitas vezes são inconsistentes entre si, já que o aparelho receptor não necessariamente sabe qual o canal de transmissão ele está recebendo dos três. Também cita como elementos disruptivos principais obstruções de 'line of sight', como objetos físicos e pessoas, e interferências de rádio como regiões com um alto número de pontos de acesso Wi-Fi, celulares e outros aparelhos que usem de radiofrequência.

É importante frisar que a escolha da avaliação da técnica de fingerprinting para resolver localização de usuário móveis em áreas demarcadas por beacons BLE em um contexto de um exergame móvel outdoor não se deve ao fato de ser a melhor técnica para a situação (até porque a mesma claramente tem suas limitações e problemas, como descrito anteriormente), e sim como prova de conceito devido a um contexto oportuno. Uma técnica mais eficaz seria, por exemplo, alguma que usasse um método probabilístico para a escolha de candidatos a posição dentro de uma área, como descrito por Honkavirta et al. [81], porém, esta teria uma implementação mais custosa, o que não ia de encontro com o objetivo principal do trabalho. A escolha tecnológica de BLE foi principalmente devido à alta disponibilidade no mercado (heterogeneidade de aparelhos e fornecedores), baixo custo, compatibilidade com uma gama de aparelhos móveis e smartphones, fácil implementação e durabilidade (bateria). Já a escolha estratégica de fingerprinting foi devido á imutabilidade de um ambiente fixo em que um jogo ocorreria (no contexto de um evento ou sessão de exergame móvel, o preparo da região em conjunto com a amostragem da fase off-line não é um problema) e a relação caótica entre a distância receptor/transmissor e as informações recebidas dos beacons (justamente dada a heterogeneidade dos tipos de aparelhos), o que dificultaria o uso de trilateração/triangulação.



### 5.2.1.
### Fases Online e Offline

Küpper [79 – Cap. 9, pág. 235 a 239] descreve, em seu livro sobre serviços de localização, fingerprinting como sendo uma técnica de posicionamento indoor em que a posição de um usuário é determinada comparando amostras de RSSI de pontos de acesso (levando em consideração Wi-Fi como tecnologia capacitadora) com uma tabela de valores pré-determinada em uma fase off-line anterior. A

estratégia usada para determinar o valor da tabela que a amostra mais se aproxima depende da precisão desejada, variando em velocidade de obtenção e complexidade de implementação. A tecnologia usada, seja pontos de acesso Wi-Fi ou beacons BLE, ZigBee ou UWB não interfere no conceito de fases do fingerprinting.

Durante a fase off-line, cada região demarcada por beacons é amostrada para a construção do mapa de rádio, que terá valores médios de intensidade dos sinais de todos os beacons que demarcam a região para cada ponto de interesse. Essa fase tipicamente só precisa ocorrer uma vez para cada sub-região demarcada antes da sessão do jogo. Partimos do princípio de que os elementos disruptivos, como quantidade de pontos de acesso Wi-Fi e número de jogadores (obstáculos de line of sight para os beacons) não mudarão durante a sessão. O número de medições necessárias dependerá da cardinalidade da área demarcada, caso ela seja em grade (que é o caso neste trabalho).

A fase online é a execução da aplicação que faz uso da estratégia de posicionamento, no caso um exergame móvel. Nela, o smartphone do usuário detecta a cada intervalo de tempo beacons próximos, guarda a informação de RSSI recebida de cada beacon e, quando tem uma amostra de tamanho suficiente, envia-a para o servidor, este que guardará a amostra com a mais recente daquele usuário. Quando o smartphone do jogador enviar um pedido de localização na sub-região através da API de localização, que será detalhada no Capítulo. 6, o mesmo usará a última amostra recebida daquele usuário e comparara-a com os valores na tabela de rádio através de alguma estratégia ou algoritmo probabilístico, retornando ao smartphone apenas a informação desejada, seja ela a posição na grade da sub-região ou quaisquer outras disponíveis na API.



### 5.2.2. Topologia dos nós (beacons)

Reiter [80] discute formas otimizadas de distribuir os nós em uma rede sem fio de forma a cobrir adequadamente uma região dependendo da tecnologia usada. Reiter comenta que Bluetooth e BLE são mais indicados a serem usados em uma topologia de estrela ou P2P com poucos nós, devido ao seu curto alcance,

enquanto ZigBee e 6LowPAN[3] são indicados para uma topologia de malha ou estrela, podendo escalar para milhares de nós. Chu et al. [68] usam uma topologia em estrela com sensores MICA2 com frequência abaixo de 1GHz, comentando que a escolha foi ideal dada a estratégia de sobreposição de células usada. Sun [35] comenta sobre a importância de um maior estudo sobre topologia de redes de nós sem fio tem para sistemas de localização dado o impacto da escolha sobre a estratégia de localização usada e que há pouco estudo na área.

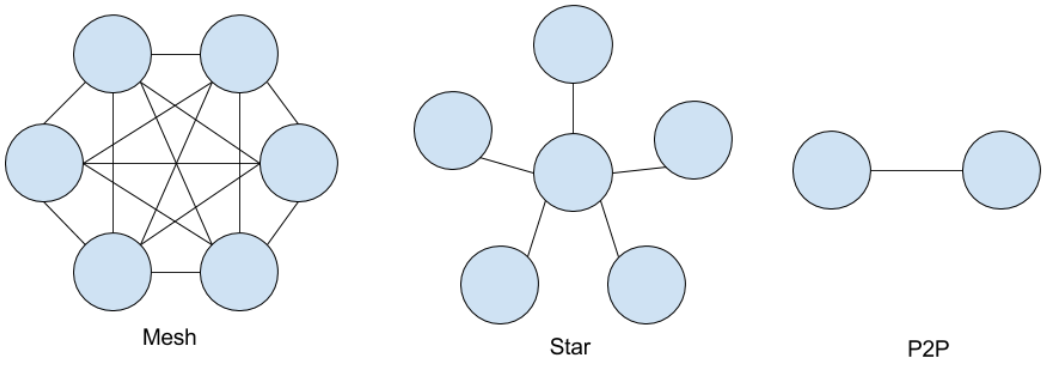


Figura. 5.2: Algumas topologias, malha, estrela e p2p, como apresentado em [80]

Para este trabalho, foi escolhida uma topologia em *estrela* como vista na Figura 5.2, não somente por ser uma das recomendadas para nós beacon BLE como por cobrir melhor uma área considerando os raios de ação de cada beacon, como pode ser visto na Figura 5.3. Usando uma topologia de malha, teríamos ou uma 'área morta' no centro da sub-região demarcada ou sobreposições em excesso, o que seria um desperdício.

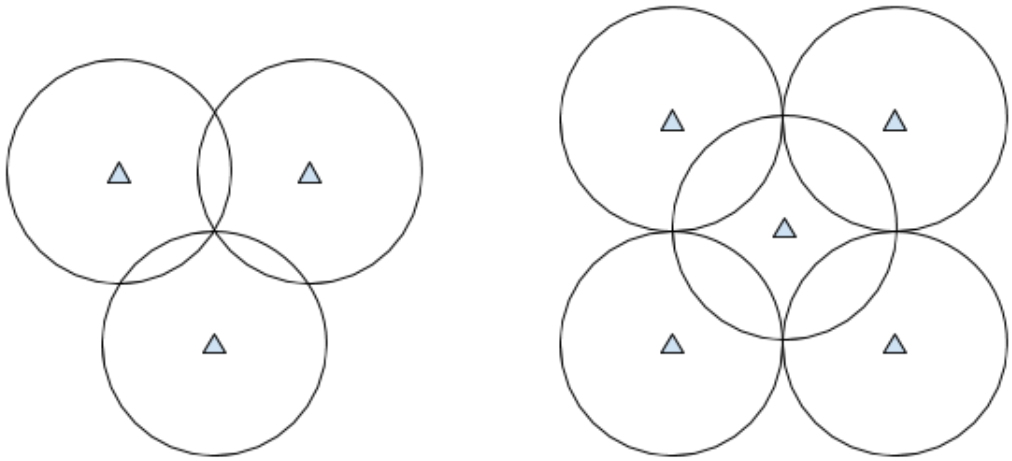

Figura. 5.3: Demarcação de duas áreas, com 3 e 5 beacons.

[3] Mais informações em http://processors.wiki.ti.com/index.php/Contiki-6LOWPAN

### 5.2.3.
### Tabela de Amostras (Radio Map)

A tabela de amostras, ou mapa de rádio, é construída na fase off-line com as amostras coletadas. Para este trabalho, essa tabela é construída levando em consideração a média e a variância das amostras recebidas. Essas informações serão comparadas com amostras menores na fase online.

A cardinalidade da tabela é a mesma da grade definida para a sub-região demarcada. Como pode ser visto no Algoritmo 1, a atualização da tabela de amostra é feita a cada nova amostra recebida. Após a amostragem de uma determinada célula da grade da sub-região, temos uma média e variância de um número de amostras de RSSI para cada beacon BLE 'visível' ao usuário na posição específica.

**Algorithm 1** Atualiza a tabela com uma nova amostra

```
1: procedure SAMPLETABLEUPDATE(PlayerSampleBuffer, gridId, cellId)
2:     RadioMap = GetRadioMap(gridId)
3:     Sample = PlayerSampleBuffer
4:     cellData = RadioMap.GetCell(cellId)
5:     for each int beaconRssi in Sample do
6:         cellData.UpdateBeaconValues(beaconRssi)
7:     end for
8:     RadioMap.UpdateCell(cellData)
9: end procedure
```

Algoritmo 1: Atualização do mapa de rádio de amostras

A atualização em si dos valores de média e variância é feita simplesmente adicionando o novo valor aos valores anteriormente guardados e recalculando.

$$variancia^2 = \frac{\sum_{i=1}^{n}(amostra_i - media)^2}{n} \tag{5-1a}$$

$$media = \frac{\sum_{i=1}^{n} amostra_i}{n} \tag{5-1b}$$

### 5.2.4.
### Métodos para comparação de amostras

Vários dos trabalhos citados nas seções anteriores descrevem estratégias e algoritmos para comparação das amostras da fase online com as do mapa de rádio. Küpper [79] comenta em seu livro que uma forma determinística de comparar amostras é através de *distância euclidiana*. Faragher [72] usa o resultado da distância euclidiana de suas amostras de RSSI BLE como variável de entrada para sua distribuição probabilística bayesiana. Honkavirta et al. [81] descrevem vários métodos determinísticos e probabilísticos, comentando que os métodos probabilísticos tiverem resultados melhores que os determinísticos em geral, e que o aumento do número de amostras a partir de um limiar não impactava na precisão alcançada.

Não é o foco deste trabalho analisar diferentes soluções e métodos determinísticos e probabilísticos para definir o mais ideal para o contexto de aplicação e tecnologias escolhidas. Trabalhos como o de Honkavirta et al. [81] e o de Sun [35] já fazem boas análises comparativas desses métodos. Também não é almejado obter um resultado melhor do que o encontrado na literatura, mas sim propor uma solução que se adeque a estratégia de fingerprinting, seja simples de implementar e interpretar os resultados, tenha execução rápida (para que seja realizável no contexto de um exergame móvel) e não necessite de nenhum dado externo não facilmente alcançável com no contexto tecnológico definido.

Sendo assim, foi escolhido que as amostras da fase online seriam comparadas as do mapa de rádio através de distância euclidiana simples (Equação 5-2), sendo escolhida como posição aproximada do jogador a célula da grade em que o menor valor da distância quadrada (para variância e média, para cada beacon) é obtido.

$$distancia = \sqrt{\sum_{i=1}^{n} \frac{(valorAmostra_i - valorTabela_i)^2}{n}} \qquad (5\text{-}2)$$

## 5.3.
## Implementação

Como já descrito, o método de fingerprinting usando beacons BLE e uma comparação de amostras da tabela de rádio determinística através de distância euclidiana foi a estratégia de localização implícita escolhida para ser implementada nesse trabalho. Essa estratégia é usada em sub-regiões de sessões de exergames móveis demarcadas por beacons BLE, que se encontram dentro de uma área de jogo. Os componentes do jogo, sendo jogadores, objetos físicos e as próprias sub-regiões tem seus estados e informações de localização (coordenadas latitude/longitude) processadas por um servidor de jogo (que executa, por exemplo, o Middleware proposto). Os jogadores carregam Smartphones atuantes como receptores de sinais dos beacons BLE através do M-Hub [66], assim como processadores locais das informações de jogo e input do jogador (dependendo do jogo) e de contexto (acelerômetro e GPS por exemplo). Estas informações de estado dos componentes do jogo são transmitidas ao servidor de jogo, que as usa para determinar condições de vitória e término do mesmo.

O uso dessa estratégia nesse contexto de aplicação foi baseado em três pilares: O uso de beacons BLE; o uso de Smartphones para interação com os beacons e com um servidor de jogo; e a integração com a API de localização.

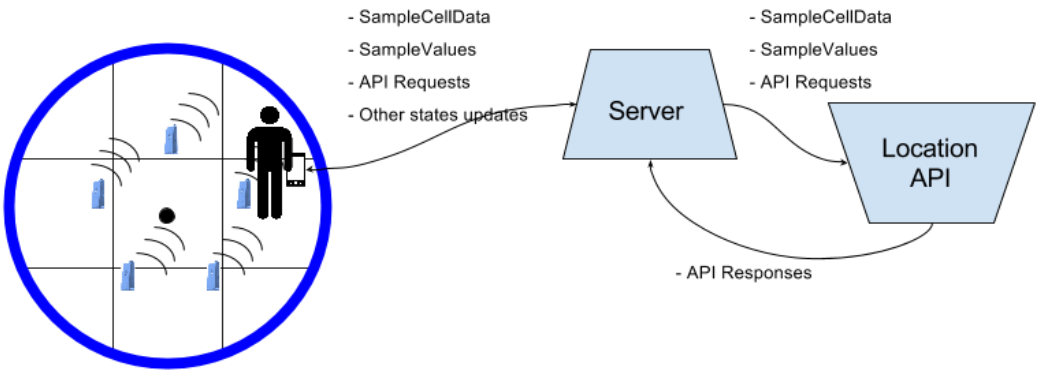


Figura 5.4: Os beacons BLE transmissores tem seus sinais recebidos por Smartphones, estes que conseguem usa-los para enviar dados para amostragem das fases off-line e online de fingerprinting, requisições da API de localização e outras atualizações de estados de componentes de jogo.

### 5.3.1.
### Beacons BLE

Beacons são aparatos capazes de transmitir um ou mais sinais em uma ou mais frequências sem necessariamente ter informação de quem os ouve. No caso dos Sensor Tags [82] da Texas Instruments[4] usados nesse trabalho (Figura 5.5), esses aparelhos são, além de beacons, sensores de temperatura, acelerômetro e proximidade. Esses aparelhos foram escolhidos pois este trabalho originalmente analisaria interações de sensores e jogadores levando em consideração as informações de contexto captáveis pelos sensores. Considerando o uso de beacons BLE devido à facilidade de integração com Smartphones e a disponibilidade, decido-se manter os Sensor Tags, já que os mesmos podem ser também usados como Beacons.

Figura 5.5: O Texas Instruments CC2541 Sensor Tag é primariamente um sensor multiuso, porém também é usável como um Beacon BLE

### 5.3.2.
### Integração com o Middleware de Exergames Móveis

O middleware proposto possibilita a implementação da estratégia de fingerprinting, como apresentado na Figura 5.4. O servidor de jogo age como intermediador para a API de localização, enquanto que o Smartphone é capaz de comunicar-se com os Sensor Tags através do M-Hub ou simplesmente captar o RSSI e identificar de quais Beacons a o sinal procede para, então, “empacotar” essa informação para o servidor usar, seja durante a fase off-line ou online.

[4] Mais informações em http://www.ti.com/lit/ml/swru324b/swru324b.pdf

### 5.3.3.
### Integração com a API de Requisições e Eventos

A API de eventos é o componente responsável pela estratégia de localização. É nela que se encontra a tabela de rádio vinculada a cada sub-região de cada sessão de jogo, instanciada pelo servidor administrativo. As chamadas e eventos de API serão melhor descritos no Capítulo 6.

A API sempre espera o recebimento de "pacotes" de amostras de beacons, independente da fase que a sessão de jogo se encontra. Caso seja um pacote da fase off-line, a API vai atualizar a mapa de rádio para a sub-região e sessão de jogo correspondente. Caso seja durante o jogo, o servidor de jogo tipicamente irá incluir as medidas recebidas dos jogadores em 'buffers' próprios para cada um deles. Assim, ao receber algum pedido de API do jogador, o servidor o encaminhará junto com as informações guardadas no buffer do jogador, assim como referência de qual sessão de jogo ele se encontra. Esse fluxo está explicitado na Figura 5.6.



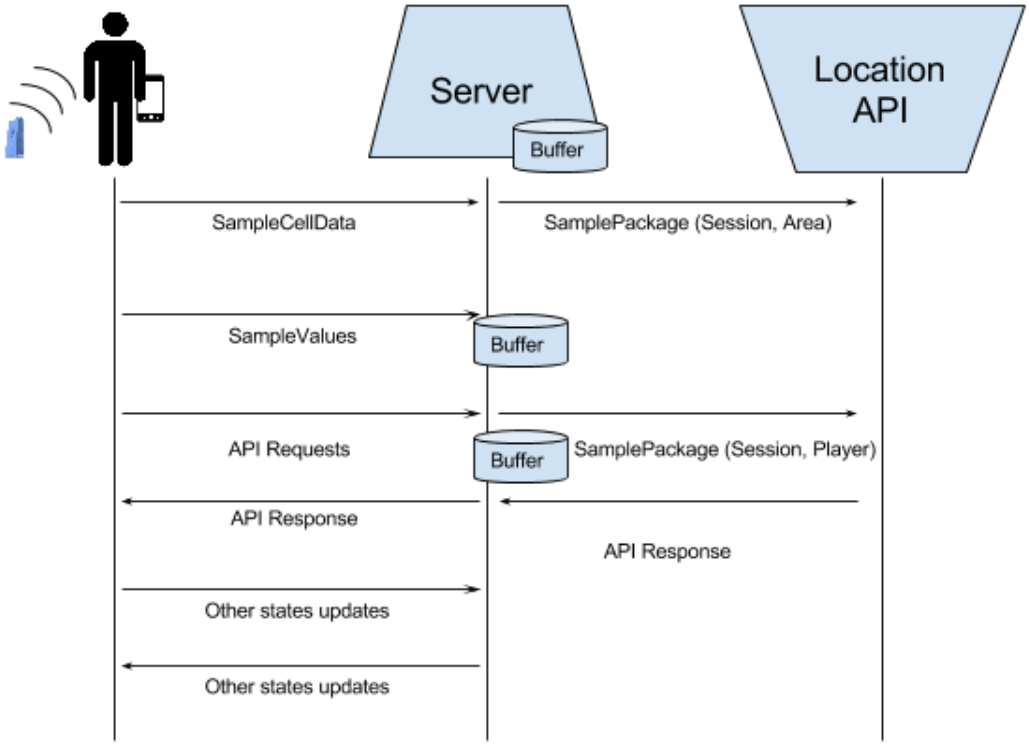


Figura 5.6: Fluxo de troca de mensagens da API relativos ao envio de amostras do jogador para o servidor de jogo

### 5.3.4.
### Sobre latência e tempo de resposta

Além da análise de precisão de localização, tem grande relevância uma análise do tempo de resposta e latência desde o envio do pacote de amostras do Smartphone do jogador até a volta da informação de localização a ele. Para esta análise, temos que considerar três fatores: O tempo de captura das amostras por parte do Smartphone, o tempo de ida e volta da mensagem através do Middleware de comunicação do SDDL e o tempo de processamento dos dados das amostras por parte do Servidor.

Como citado no Capítulo 4, o tempo aproximado de ida e volta de uma mensagem unicast, contando o tempo de processamento, contendo a informação de localização por parte do SDDL é de no máximo 50ms. O tempo do cálculo de uma distância euclidiana, que é a estratégia de comparação de amostras escolhida, é de menos de 1ms para centenas de execuções, de acordo com testes realizados com a linguagem Java (em que o Middleware e a API são escritos). Já o tempo de captura das informações de amostras é definido por configuração interna do M-Hub, podendo ter qualquer valor desejado, mas idealmente ficando em uma janela de 100ms. Isso significa que a cada 100ms, capturamos novas informações de amostras, podendo assim envia-las ao Servidor em tempo hábil.

Nestas condições, considerando ainda uma latência de rede e o sistema sob estresse (como no caso dos testes do SDDL citados no Capítulo 4), o jogador recebe informação de sua localização em aproximadamente 200ms. Além disso, como o envio de amostras é constante e a análise das amostras é realizada em fila no servidor, o jogador sempre receberá informações novas a respeito de sua localização, mesmo que atrasada em um ou dois ciclos de 200ms. Assim, dada a construção do fluxo, não ocorre indisponibilidade de informação de localização para o jogador.

Esses valores são suficientes para o contexto de exergame móvel, dado a definição de ritmo acelerado e a necessidade de informações atualizadas a cada segundo para os jogadores terem uma experiência correta de jogo.

## 5.4. Avaliação

A estratégia de localização foi avaliada através de testes de campo. Cada teste consistia em um usuário se movimentando dentro de sub-regiões demarcadas em diferentes configurações enquanto carregam Smartphones. Antes dos testes, o usuário realiza a fase off-line do fingerprinting de modo a enviar as amostras de sinais para o servidor e construir o mapa de rádio.

Para o padrão de movimento do usuário tanto na fase off-line quanto online, foram considerados ideais padrões de movimento que simulassem a movimentação esperada de um jogador em um exergame móvel, porém que fosse repetível, para comparação futura de resultados. Assim, foram realizados testes com padrões realistas (de ritmo acelerado, aleatórios, imprevisíveis) e não realistas (devagar, previsível, linhas retas) para o contexto de jogo de forma a analisar o impacto nos resultados.



### 5.4.1. Configuração dos Testes

As seguintes configurações variáveis foram realizadas para diferentes testes:

- Sub-regiões com números diferentes de beacons (3 e 5);
- Áreas com diferentes densidades de pontos de acesso Wi-Fi próximos (poucos e muitos);
- Sub-regiões com diferentes números de jogadores interagindo (somente o usuário de teste e dois jogadores extras);
- Sub-regiões em que os beacons encontram-se posicionados de maneiras diferentes (no chão e em um pedestal na altura do jogador);
- Sub-regiões com diferentes cardinalidades de grade (2x2 e 3x3);
- Com diferentes padrões de movimento durante a amostragem (parado olhando para o centro e variando em quatro direções);
- Com diferentes padrões de movimento durante o teste (em linha reta, devagar e aleatoriamente, rápido).

Além das configurações variáveis, usamos as seguintes configurações constantes para todos os testes:

- Raio da sub-região: 5.2 metros

- Raio de ação de cada beacon: 1.5 metro
- Vinte amostras para cada célula durante a fase off-line;
- Três amostras de buffer dinâmico para cada jogador;
- Janela de 500ms para scan de beacons BLE por parte do Smartphone;
- Intervalo de 500ms entre janelas;
- Smartphone ASUS Zenfone 2 ZE551ML conectando-se com 4G LTE no servidor de testes;
- Servidor de testes executando em cloud no Microsoft Azure Standard_A0\ExtraSmall com 1 core CPU, 768MB de RAM e disco de 20GB com 500 IOPS.

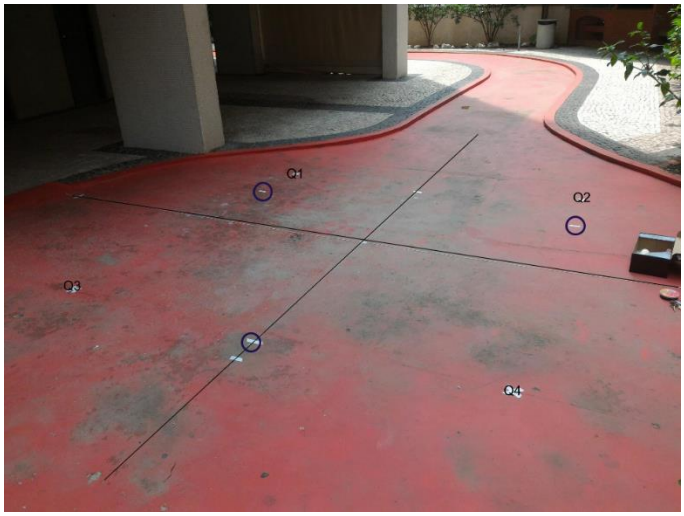


Figura 5.7: Região usada para testes, com raio de 5.2 metros

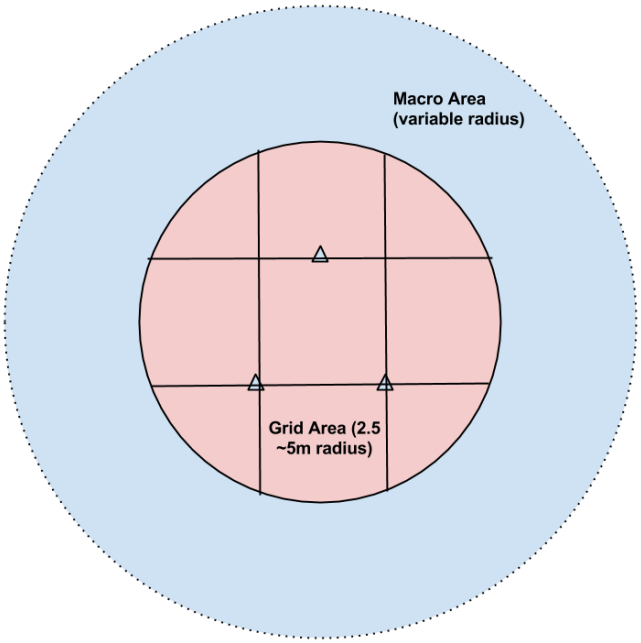


Figura 5.8: Exemplo de área demarcada por 3 beacons com uma cardinalidade 3x3. A área externa pode conter outras sub-regiões com diferentes configurações.



### 5.4.2.
### Elementos disruptivos

Como citado por Faragher [72] e outros, dois dos maiores elementos disruptivos esperados em um sistema baseado fingerprinting são o número de obstáculos bloqueadores de linha de visão (line of sight) e as interferências de rádio. De forma a quantificar esses elementos para os testes, consideramos testes com apenas o testador/jogador na sub-região e com mais duas pessoas ativamente bloqueando os beacons, assim como testes em regiões com muitos pontos de acesso Wi-Fi (aproximadamente 55) e com poucos (aproximadamente 10).

### 5.4.3.
### Interpretação dos Resultados

Foram realizados testes de forma a cobrir todas as possibilidades de configuração definidas, focando nos casos em que os padrões de movimento do testador se aproximavam dos esperados em um exergame móvel. Além disso, foram considerados dois intervalos de precisão para cada cardinalidade de grade, um para resultados que cobrissem exatamente a célula desejada (0.85m para 3x3 e 1.3m para 2x2) e outro que também considerasse as células vizinhas (1.7m e 2.0m

respectivamente). Chamaremos essas duas precisões de alta e baixa para melhor entendimento.

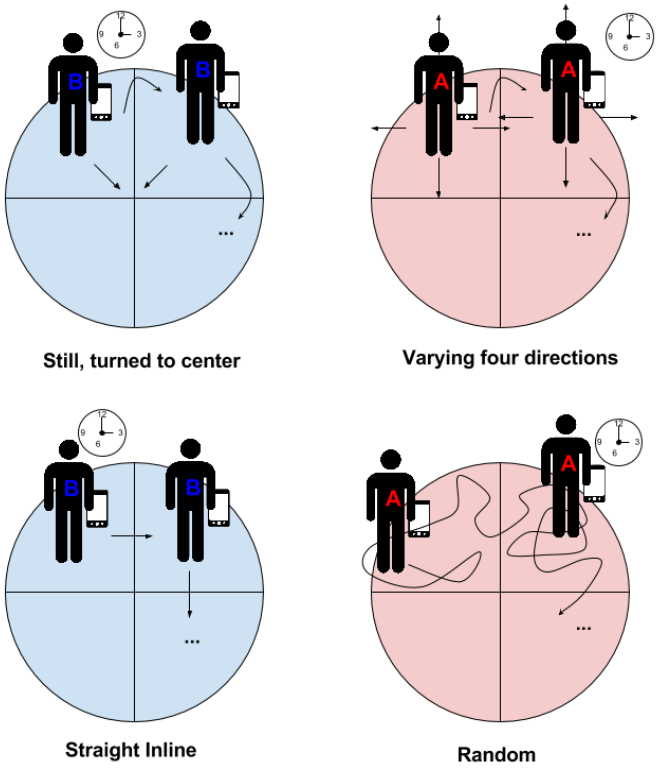


Figura 5.9: Os dois padrões de movimento testados para cada uma das fases: Parado em direção ao centro e variando em quatro dimensões para a off-line, andando em linha reta e aleatoriamente para a online.

Os resultados médios foram dentro do esperado, considerando as perturbações relativas a interferências de rádio e obstruções, a diferença média de sinal emitido de cada beacon devido a durabilidade da bateria [83] e o próprio algoritmo de comparação de amostras escolhido, mais fraco do que outras opções não determinísticas [35, 81].

| Grid/Precisão | Taxa de acerto |
|---|---|
| 3x3/0.85m | 32.4% |
| 3x3/1.70m | 72.8% |
| 2x2/1.30m | 56.1% |
| 2x2/2.00m | 91.6% |

Tabela 5.1: Taxa de acerto por precisão e cardinalidade.

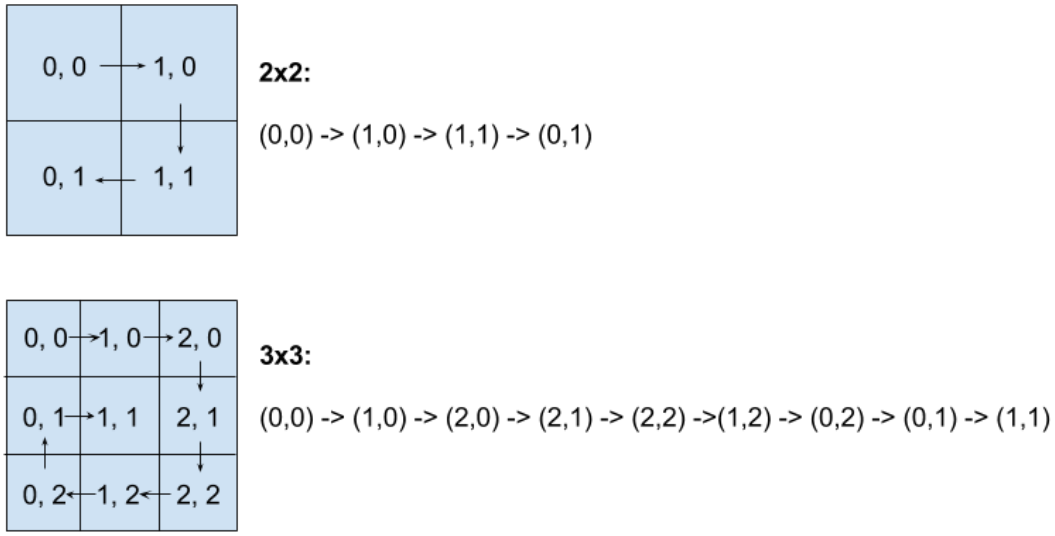


Figura 5.10: Caminho percorrido nas células durante as fases off-line e online para as duas cardinalidades de grade de sub-região.

- Sobre os pontos de acesso Wi-Fi:

Foram realizados testes em três regiões com densidade de pontos de acesso diferentes, uma com 10 a 15 pontos (baixa), outra com aproximadamente 40 pontos (média) e outra com 55 ou mais pontos (alta). A precisão do mesmo teste realizado em uma região de baixa para média densidade caiu 7% para baixa precisão e 9% para alta, enquanto que de média para alta densidade caiu 14% para baixa e até 42% para alta precisão.

- Padrão de movimentação durante a fase off-line:

Era esperado que, ao adicionar complexidade no padrão de movimento, haveria um efeito negativo na precisão. Isto de fato ocorreu, apesar do efeito diminuto quando o jogador de teste tinha padrões de movimentação semelhante nas duas fases. Quando o jogador passou a fazer a amostragem variando as quatro direções para cada célula ao invés de ficar imóvel na direção do centro da sub-região, a precisão diminuiu em 12% para testes em linha reta de baixa precisão e 9% em alta. Já para movimento aleatório, a precisão diminuiu apenas 3% nos dois casos.

- Padrão de movimentação durante a fase online:

Também como esperado, quando o jogador se movimenta com maior liberdade na grade, em várias direções e com maior velocidade, a precisão é menor do que quando apenas em linha reta (-16% para baixa precisão e -22% para

alta). Apesar disso, quando acompanhada de amostragem também variando direções, o impacto é menor (-2% para baixa e -18% para alta).

- Altura dos Beacons:

Ao colocarmos os Beacons em uma altura mais próxima da que os jogadores carregam os seus Smartphones, houve uma ligeira melhora na precisão (+10% para baixa precisão e +7% para alta).

- Cardinalidade da grade:

Quando aumentada a cardinalidade da grade de 2x2 para 3x3, a precisão de acertos diminui drasticamente tanto para baixa (25%) quanto para alta precisão (47%).

- Número de beacons BLE:

Por outro lado, ao aumentarmos a densidade de beacons demarcando a sub-região, obtemos um aumento razoável na precisão (36% para baixa precisão, 9% para alta).

- Obstruções de linha de visão:

Ao adicionarmos uma pessoa ativamente obstruindo os beacons da sub-região do jogo, ao contrário do esperado, não houve piora tão significativa na precisão (10% de piora para baixa precisão e inconclusivo para alta).

Analisando em mais detalhe os resultados, percebemos que casos que o padrão de amostragem se assemelhava ao de teste (“parado olhando para o centro” + “reto em linha reta” e “variando em quatro direções” + “aleatório”) tiveram resultados melhores do que quando do contrário. A relação faz sentido já que, por definição, o mapa radial deve ter valores aproximados para cada região de forma que o valor de cada célula tenha uma relação (determinística ou probabilística) com os valores recebidos na fase online. Também conseguimos confirmar algumas suspeitas prévias, que o aumento da cardinalidade da grade causava uma piora na precisão, já que os resultados corretos passaram a ter um raio menor (vide Tabela 5.1), e também que o aumento da densidade de beacons mapeando a sub-região causa um aumento na precisão.

Além disso, ao posicionarmos os beacons em um pedestal para que sua altura fique mais próxima dos Smartphones carregados pelos usuários, ao invés de deixá-los no chão, houve uma ligeira melhora na precisão. Isso provavelmente deve-se ao fato da linha de visão se tornar menos obstruída quando o usuário não

está andando por cima dos beacons. Tudo indica que, posicionando-os em uma altura com possivelmente ainda menos bloqueios por outros jogadores e objetos, como no teto, a precisão aumentaria ainda mais. Também como previsto, adicionar uma pessoa obstruindo ativamente os beacons durante fase online causou uma piora na execução. Apesar disso, essa piora não foi significativa, possivelmente por se tratar de apenas uma pessoa.

Se tratando de tempo de resposta, durante o teste, como definimos uma janela de busca de 500ms e um envio periódico das amostras também a cada 500ms, temos uma ida-e-volta de informação de aproximadamente 600ms, com um eventual ciclo de amostras repetido devido a janela de detecção demasiadamente grande. Como definimos que respostas dentro de um segundo são aceitáveis, apesar da configuração não ideal, ainda temos um resultado dentro do esperado.

## 5.5. Discussão

Apesar dos elementos de interferência e impeditivos, o impacto nos casos em que estas perturbações foram consideradas também durante a fase off-line foi um pouco menor. A estratégia de comparação das amostras tem um grande impacto nesse quesito, já que em um modelo probabilístico (ou um determinístico de mais poderoso, como K-Nearest), por exemplo, caso as perturbações sejam constantes na fase off-line e online, teríamos resultados ainda mais precisos [35].

Também poderíamos diminuir a janela de scan dos beacons BLE por parte dos Smartphones para 100ms, como sugerido por Faragher [72]. A janela foi definida como 500ms pois ao diminuirmos mais que isso muitos dos Sensor Tags usados não tinham sua transmissão captada, o que dificultava a execução. Esse problema é devido a relação entre potência de sinal transmitido e a bateria dos beacons, como descrito por Saxena et al. [83]. Possivelmente, usando beacons mais especializados e com baterias ‘saudáveis’, poderíamos diminuir o intervalo para 100ms e conseguir resultados melhores, melhorando também a latência total do tempo de ida-e-volta para a atualização da informação de localização do jogador.

Quanto ao problema de obstrução pelos jogadores, uma opção seria posicionar os beacons “no alto” da sub-região ou “pendurados” em algum tipo de

apoio. Geddes [84] comenta que posicionar os beacons acima da altura da cabeça dos usuários tem um efeito positivo na consistência do RSSI. Dessa forma, eles sempre estariam desobstruídos, dado que os jogadores não estão passando diretamente na frente deles em nenhum momento. Também, para obtermos resultados mais confiáveis, seria interessante que houvessem mais configurações de testes, considerando mais beacons, diferentes topologias de mapeamento, cardinalidades mais altas de grade e mais jogadores na região de jogo.

Considerando o resultado, vemos que é possível usar a estratégia de microlocalização de Fingerprinting por amostragem com beacons BLE em um ambiente outdoor característico de um exergame móvel, dado que aja uma boa estratégia de comparação de dados por trás e que certas condições de infraestrutura sejam alcançadas. Consideramos que a análise da estratégia aqui descrita para fins de prova de conceito foi bem-sucedida, pois conseguimos um resultado comparável com outras estratégias usando tanto o Middleware implementado quanto a API de microlocalização proposta.

De todo o modo, poderíamos melhorar a estratégia proposta de diversas formas. Além das melhorias de infraestrutura comentadas acima, há algumas melhorias na estratégia como um todo que poderiam ser realizadas em trabalhos futuros. De imediato, K-Nearest é uma estratégia determinística e realizável para o contexto desejável que traria resultados melhores do que distancia euclidiana pura. Seria também interessante comprovar a suspeita que modelos probabilísticos podem dar resultados melhores e se seria viável para o contexto de exergame móvel, que requer resultados em tempo ‘quase’ real[5].

Além disso, seria essencial comparar os resultados obtidos dos beacons BLE com outras tecnologias, como ZigBee e MICA2. Para isso basta que os Smartphones tenham alguma forma de receber as transmissões desses aparelhos, pois seguindo o fluxo para o servidor, não importa o tipo de aparelho de que ele recebe os valores de RSSI. Uma outra direção para trabalhos futuros seria em relação ao preparo da infraestrutura da sessão de jogo. Uma linha de pesquisa que segue para possibilidade de automatizar a demarcação das sub-regiões de microlocalização com os beacons, ou uma forma de definir a cardinalidade da

[5] Que coincida com a soma da janela de scan e do intervalo entre scans dos beacons BLE, aproximadamente 500ms por exemplo.

grade e densidade de mapeamento automaticamente a partir do ambiente de implantação seria muito adequado para uma possível continuidade deste trabalho.

# 6
# API de Requisições e Eventos de Localização

Este capítulo apresenta a principal contribuição desse trabalho, a implementação de uma API de Localização capaz de responder a requisições externas e de engatilhar eventos para componentes que os estejam ouvindo. A API existe dentro do contexto do middleware proposto do Capítulo 4 e usa a estratégia de microlocalização proposta no Capítulo 5. Através da API, Smartphones de jogadores em uma sessão de exergame móvel são capazes de fazer requisições de informações relacionadas a microlocalização de um ou mais jogadores em sub-regiões demarcadas com beacons BLE para o servidor de jogo. O servidor, então, repassará as requisições juntamente com outras informações necessárias para seu processamento para a API, que passará a um *location handler* específico da estratégia de localização a ser executada, e devolverá a resposta para o Smartphone do jogador.

Esta API visa capacitar um desenvolvedor de exergame móvel a criar diferentes jogos com as requisições e eventos disponíveis, assim como dar possibilidade de extensão para implementação de novos componentes.



## 6.1.
## Conceito de SOA dirigido a Eventos

APIs são interfaces de aplicações que abstraem certos procedimentos e requisições de dados para outros serviços, aplicações ou usuários [77]. A API de Localização é baseada em SOA e existe dentro da arquitetura idealizada para o middleware, como comentado no Capítulo 4. O servidor de jogo é capaz de se comunicar com um módulo separado que provê o serviço de localização vinculado a uma sessão de jogo. Dessa forma, dados como última localização e o buffer dos últimos valores de sinais de um jogador ficam em memória nesse módulo para, quando necessário, serem repassados para a API de localização em conjunto com os dados de uma requisição vinda do Smartphone de um jogador (Figura 6.1). As chamadas de API para as requisições disponíveis serão apresentadas mais à frente.

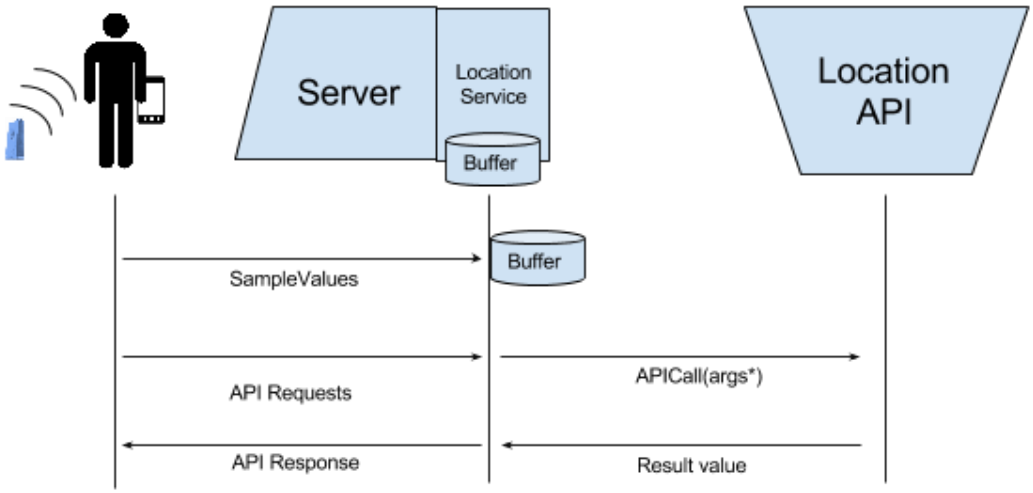


Figura 6.1: Fluxo de Requisições e Resposta da API de Localização

Além das chamadas de API para requisições, podemos vincular eventos de localização às sessões de jogo. Os eventos são transmitidos a todos os jogadores, podendo a implementação do jogo no Smartphone do jogador usar desta informação quando recebida ou não. O fluxo do engatilhamento de eventos pode ser visto na Figura 6.2.

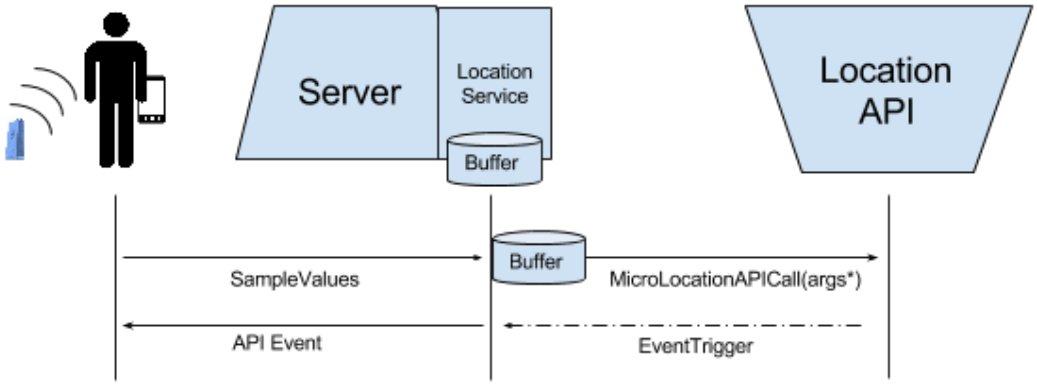


Figura 6.2: Fluxo do engatilhamento de eventos. Não é sempre que eventos são engatilhados após a definição da localização mais recente de um usuário, apenas quando as condições de cada evento são atingidas.

O fato da arquitetura dar suporte ao engatilhamento de eventos por parte da API de Localização torna-a mais próxima do padrão de SOA dirigido a eventos, ou SOA 2.0 [85], que nada mais é do que SOA com a inclusão de conceitos de uma arquitetura dirigida a eventos para troca de mensagens entre seus componentes. Os dados trocados entre os componentes da arquitetura ainda

são no formato de serializáveis e de stream de dados, assim como comentado no Capítulo 4 devido ao uso do middleware de comunicação do SDDL, o que difere do comumente praticado em arquiteturas SOA.

## 6.2. Trabalhos Relacionados

Foram encontrados alguns trabalhos cujas contribuições apresentadas são ou se aproximam de interfaces ou abstrações para serviços especializados relacionados a jogos pervasivos, aplicações AR ou moveis sensíveis a contexto.

Em FinN [39], o framework proposto inclui gerenciamento de eventos de usuário no fluxo contrário do proposto neste trabalho. Lá, na camada "Guardiã", em que os aparelhos móveis dos jogadores estão inseridos, qualquer ação tomada por um jogador é considerada um evento que deve ser repassado para o servidor do jogo. Quando recebido, o servidor lida com esses eventos engatilhados da maneira correta para o contexto de jogo implementado. Um comparativo entre quem detém a responsabilidade de engatilhar eventos de jogo entre este trabalho e FinN pode ser visto na Figura 6.3.

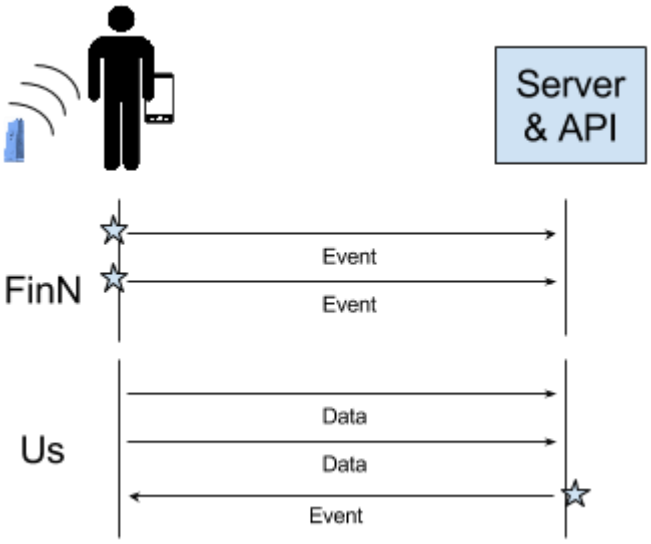


Figura 6.3: Comparativo entre FinN e a solução proposta. Ocorre uma inversão de responsabilidade quanto a quem engatilha o evento (estrela).

A geração de eventos no fluxo Smartphone-Servidor pode ser interessante para o processamento de regras em cima dos eventos gerados por cada jogador. A

possibilidade de mesclar estes fluxos é sugerida como trabalho futuro mais à frente.

O framework Morgan [59] para projetos de VR inclui uma biblioteca escrita em C++ que implementa a Morgan API para requisições relacionadas a VR e AR. A API abstrai métodos e requisições do framework e reduz a complexidade para desenvolvedores de aplicações. Métodos incluem acesso a informações dos nós do grafo de cena independente do motor de renderização usado e manipulação dos nós (inserção, deleção, cópia, etc.).

Em MobiCon, plataforma de monitoramento de contexto móvel proposta no trabalho de Lee et al. [76], são inclusas várias APIs genéricas para monitoramento de contexto com o objetivo de facilitar a delegação de tarefas complexas de monitoramento de contexto por parte das aplicações construídas na plataforma, de forma que essas aplicações possam focar em suas logísticas próprias.

Por último, no trabalho de Hwang et al. [78], também é proposta uma plataforma de jogos móveis, a *Player Space Director*, que expõe uma API que abstrai de aplicações diversos tipos de chamadas, requisições e manipulação de dados relacionados a processamento de contexto, gestos, balanceamento e regras de jogo e suporte à interação remota (Figura 6.4).

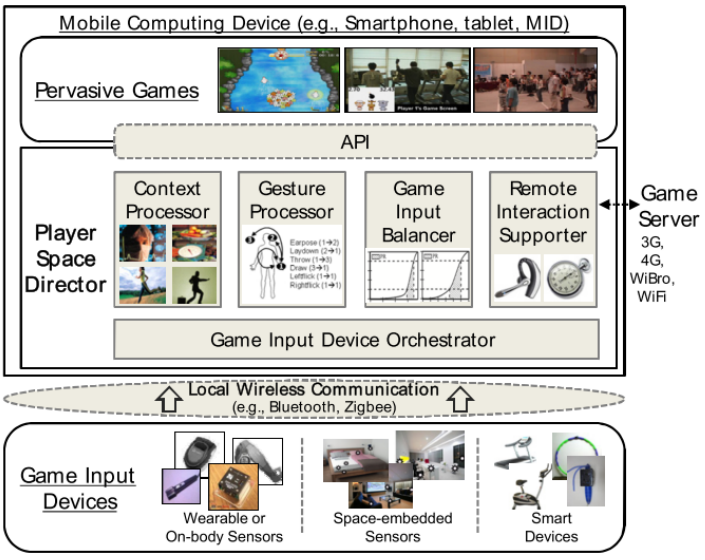


Figura 6.4: Arquitetura da plataforma PSD, como vista em [78], com a API exposta para aplicações de jogos pervasivos serem construídas.

As APIs dos trabalhos analisados consistem em partes de sistemas maiores, frameworks e plataformas em geral. Como comentado no Capítulo 4, o middleware em que a API está inclusa, como um todo, não foi desenvolvido como foco principal desse trabalho. Apesar disso, direcionar uma linha de pesquisa futura para estender o trabalho proposto em um framework para exergames móveis pode render bons frutos, como será discutido no Capítulo 7.

## 6.3. Detalhes da Implementação

Nesta seção, serão discutidos detalhes de implementação da API e sua integração com o restante do middleware.

### 6.3.1. Componentes do Sistema

Na parte do servidor de jogo, como apresentado na Figura 4.5 e 4.7 do Capítulo 4, a API é exposta para um *Location Resolver*, que nada mais é do que o *Location Service* que guarda as informações de localização referentes a uma sessão de jogo. Através dele, chamadas de *doRequest(*args)* são realizadas para a API, que resolverá a requisição dependendo da estratégia de localização requerida através de um *Location Handler*. No caso, foi implementado um Location Handler específico para a estratégia apresentada no Capítulo 5 (Fingerprinting, beacons BLE, mapa de rádio com comparação determinística de amostras por distância euclidiana pura). Dessa forma, outros location handlers poderiam ser implementados para que a microlocalização fosse determinada utilizando outras estratégias.

Da parte do cliente móvel executando no Smartphone dos jogadores, as informações de quais métodos a API expõe para este jogo específico é descrita em uma interface contida no código do jogo, assim como o servidor sabe quais são os métodos da API que os jogadores de um determinado jogo têm acesso. Este ponto é importante, pois assim como um jogo pode necessitar da implementação de diferentes estratégias de localização, executadas por diferentes location handlers, um mesmo servidor pode estar executando sessões de diferentes tipos de jogos. Assim, apesar do vínculo das informações de requisições de API e informações de localização dos componentes de jogo ser uma sessão para um

location service, o que a API expõe é definido pelo tipo de jogo, e não por sessão (Figura 6.5).

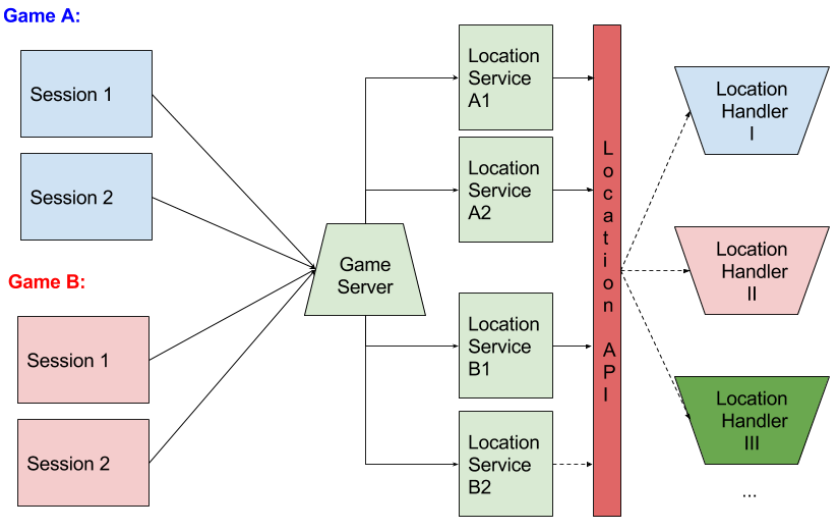


Figura 6.5: Vínculo entre jogadores em uma sessão de jogo e os serviços de localização são independentes de qual é o jogo sendo executado. O tipo de jogo pode requisitar diferentes estratégias de localização, estas executadas pelo Location Handler correspondente.

### 6.3.2. Configuração dos clientes móveis

Os exergames móveis a serem desenvolvidos no paradigma proposto são parcialmente executados como aplicações móveis na plataforma Android. Apesar das requisições à API de localização feitas por eles serem assíncronas e variarem dependendo do jogo desenvolvido, as informações de contexto necessárias para a determinação da microlocalização são enviadas do cliente ao servidor sincronamente com intervalos padrão fixos, assim como pedidos de microlocalização opcionais. Em se tratando da implementação atual, dado que a janela de busca por beacons mais o intervalo entre buscas é de 500ms, o envio das informações de sinais a serem salvas no buffer do serviço de localização são enviadas a cada 600ms, de modo a sempre serem informações da última busca realizada. Esses intervalos são configuráveis por estratégia de localização (atualmente, existindo somente uma, a proposta no Capítulo 5).

## 6.4.
## Elementos da API

Os elementos da API são as requisições e eventos expostos pela mesma para o serviço de localização. Nesta seção apresentaremos estes elementos que foram implementados neste trabalho.

### 6.4.1.
### Requisições

As requisições implementadas para este trabalho possibilitam que um exergame móvel executando em um Smartphone de um jogador consiga se informar sobre a localização de um jogador em uma grade de uma sub-região do jogo, se existe um outro jogador co-localizado na mesma célula de uma grade, quanto tempo o jogador se encontra na mesma célula, a distância aproximada entre dois jogadores em uma mesma sub-região e se o jogador se encontra em uma célula ou sub-região especifica.

A requisição base da API é chamada de *MicroLocation* (Algoritmo 3). Através dela, é feita a consulta no mapa de rádio das informações de amostras guardadas para definir onde o jogador se encontra, a partir de informações do buffer de sinais dele. Internamente, o método chama *LocateGridFromSample* (Algoritmo 2), capaz de descobrir em qual sub-região o jogador se encontra com base em quais beacons são visíveis a ele (independente do RSSI). Todas as requisições fazem uso do id da sessão em que o jogador se encontra, dado que o próprio serviço de localização é vinculado à sessão de jogo.



**Algorithm 2** Descobre a sub-região que a lista de beacons da amostra de sinais cobre

```
1: function LocateGridFromSample(PlayerSampleBuffer, sessionId)
2:     Sample = PlayerSampleBuffer
3:     AreaList = GetSession(sessionId).GetAreaList()
4:     for each int Area in AreaList do
5:         if Sample.BeaconList.Contains(Area.BeaconList) then
6:             return Area
7:         end if
8:     end for
9:     return null
10: end function
```

Algoritmo 2

**Algorithm 3** Descobre a célula que o jogador se encontra com base na amostra de sinais

```
1: function MicroLocation(PlayerSampleBuffer, sessionId)
2:     Sample = PlayerSampleBuffer
3:     Area = LocateGridFromSample(Sample, sessionId)
4:     RadioMap = GetRadioMap(Area)
5:     cellId = RadioMapQuery(RadioMap, Sample)
6:     return cellId
7: end function
```

Algoritmo 3

Duas outras requisições são executadas pelos métodos *IsInCell* (Algoritmo 4) e *IsInGrid* (Algoritmo 5), capazes de identificar se um jogador se encontra em uma célula ou sub-região específica.

**Algorithm 4** Verifica se o jogador se encontra em uma célula especifica

```
1: function IsInCell(PlayerSampleBuffer, cellId, sessionId)
2:     Sample = PlayerSampleBuffer
3:     currentCellId = MicroLocation(Sample, sessionId)
4:     if currentCellId = cellId then
5:         return True
6:     end if
7:     return False
8: end function
```

Algoritmo 4

**Algorithm 5** Verifica se o jogador se encontra em uma área específica

```
1: function IsInArea(PlayerSampleBuffer, areaId, sessionId)
2:     Sample = PlayerSampleBuffer
3:     Area = LocateGridFromSample(Sample, sessionId)
4:     if Area.Id = areaId then
5:         return True
6:     end if
7:     return False
8: end function
```

Algoritmo 5

A requisição de co-localização de jogadores chama o método *ArePlayersCoLocalized* (Algoritmo 6) de forma a identificar se dois jogadores estão em uma mesma célula. Como existem casos além de estarem ou não na mesma célula, o método retorna um enumerador que identifica o estado relativo entre os jogadores, podendo ser *Same*, para jogadores na mesma célula, *NotSame*, para jogadores na mesma sub-região, mas não na mesma célula e *NotSameArea*, para jogadores em sub-regiões diferentes. Não há necessidade de analisar se dois

jogadores estão na mesma sessão nesse nível de aplicação, pois isso já é verificado anteriormente ao recebida a requisição.

**Algorithm 6** Verifica dois jogadores estão co-localizados

```
1: function ArePlayersCoLocalized(PlayerSampleBuffer_1, PlayerSampleBuffer_2, sessionId)
2:     Sample_1 = PlayerSampleBuffer_1
3:     Sample_2 = PlayerSampleBuffer_2
4:     cellId_1 = MicroLocation(Sample_1, sessionId)
5:     areaId_1 = LocateGridFromSample(Sample_1, sessionId)
6:     cellId_2 = MicroLocation(Sample_2, sessionId)
7:     areaId_2 = LocateGridFromSample(Sample_2, sessionId)
8:     if areaId_1 = areaId_2 AND cellId_1 = cellId_2 then
9:         return CoLocalization.Same
10:    end if
11:    if areaId_1 ≠ areaId_2 then
12:        return CoLocalization.NotSameArea
13:    end if
14:    return CoLocalization.NotSame
15: end function
```

Algoritmo 6

A distância entre dois jogadores é calculada no método *AproximateDistance* (Algoritmo 7), sendo ela uma Distância de Manhattan entre as células em que os jogadores se encontram (variando a distância entre células dependendo da cardinalidade da sub-região).

**Algorithm 7** Retorna a distância aproximada entre dois jogadores

```
1: function AproximateDistance(PlayerSampleBuffer_1, PlayerSampleBuffer_2, sessionId)
2:     Sample_1 = PlayerSampleBuffer_1
3:     Sample_2 = PlayerSampleBuffer_2
4:     CoLocalizationResult = ArePlayersCoLocalized(Sample_1, Sample_2, sessionId)
5:     if CoLocalizationResult = CoLocalization.NotSameArea then
6:         return −1
7:     end if
8:     cellId_1 = MicroLocation(Sample_1, sessionId)
9:     cellId_2 = MicroLocation(Sample_2, sessionId)
10:    return GetManhathanDistance(cellId_1, cellId_2)
11: end function
```

Algoritmo 7

A requisição de quanto tempo o jogador se encontra na mesma célula, *RequestTimeInCell* (Algoritmo 8), utiliza informações adicionais do serviço de localização para o cálculo do tempo, como a última localização do jogador e seu tempo de entrada na célula.

**Algorithm 8** Retorna o tempo que o jogador está na mesma célula

```
1: function RequestTimeInCell(playerId, sessionId)
2:     Sample = GetPlayerSampleBuffer(playerId)
3:     lastKnownCell = GetLaskKnownPosition(playerId)
4:     timeOfEntrance = lastKnownCell.TimeOfEntrance
5:     if IsInCell(lastKnownCell, Sample, sessionId) then
6:         return Time.Now() − timeOfEntrance
7:     end if
8:     return −1
9: end function
```

Algoritmo 8

Por último, existem requisições que necessitam de um pré-processamento no próprio serviço de localização antes de serem encaminhadas para o location handler e para a API. Em *RequestIsInCell* (Algoritmo 9) e *RequestIsInGrid* (Algoritmo 10), é verificado antes se o jogador em questão já tem uma última posição conhecida recente. A chamada de API é realizada apenas caso negativo. Isso é possível pois a atualização da última posição conhecida do jogador é sempre realizada quando o buffer de valores de sinais de beacons é atualizado, assim como o tempo de atualização (temos o tempo de entrada na célula e o tempo de última atualização), podendo assim verificar se a informação é recente ou não.

**Algorithm 9** Request para IsInCell

```
1: function RequestIsInCell(playerId, cellId, sessionId)
2:     Position = GetLastKnownPosition(playerId)
3:     if Position = null then
4:         Sample = GetPlayerSampleBuffer(playerId)
5:         return IsInCell(Sample, cellId, sessionId)
6:     else
7:         lastKnownCell = Position.CellId
8:         if lastKnownCell = cellId then
9:             return True
10:        end if
11:    end if
12:    return False
13: end function
```

Algoritmo 9

**Algorithm 10** Request para IsInGrid

```
1: function RequestIsInGrid(playerId, gridId, sessionId)
2:     Position = GetLastKnownPosition(playerId)
3:     if Position = null then
4:         Sample = GetPlayerSampleBuffer(playerId)
5:         return IsInGrid(Sample, gridId, sessionId)
6:     else
7:         lastKnownGrid = Position.CellId
8:         if lastKnownCell = cellId then
9:             return True
10:        end if
11:    end if
12:    return False
13: end function
```

Algoritmo 10

### 6.4.2.
### Eventos

Eventos são registrados por sessão e transmitidos para todos os jogadores que se encontram nelas. Sua ocorrência é sempre verificada no serviço de localização a cada atualização da última localização conhecida de um jogador, esta que é atualizada a cada chamada de MicroLocation para a API. Os eventos engatilhados são repassados para o game server, que o transmite para todos os jogadores registrados na sessão que o evento ocorreu.

Foram implementados apenas três eventos, havendo ainda grande espaço para melhorias no tratamento destes. Os eventos implementados são vinculados a ocorrências relativas a sub-região e suas células, tais como quando um jogador entra ou sai de uma determinada sub-região – *GridEvent*, quando um jogador entra ou sai de uma determinada célula – *GridCellEvent*, e quando um jogador entra em uma célula em que um outro jogador já se encontra – *CoLocationEvent*. Esses eventos e suas ocorrências podem ser visto em maior detalhe na Figura 6.6.

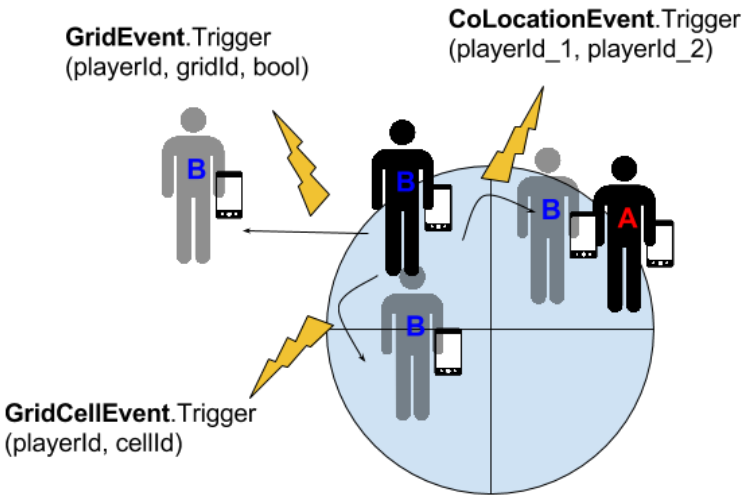


Figura 6.6: Eventos implementados da API de localização

## 6.5. Arquitetura do Serviço de Localização

O Serviço de Localização é o componente do middleware que gerencia a interface com a API, passando as informações necessárias para que ela saiba que Location Handler usar para resolver a localização (Figura 6.7).

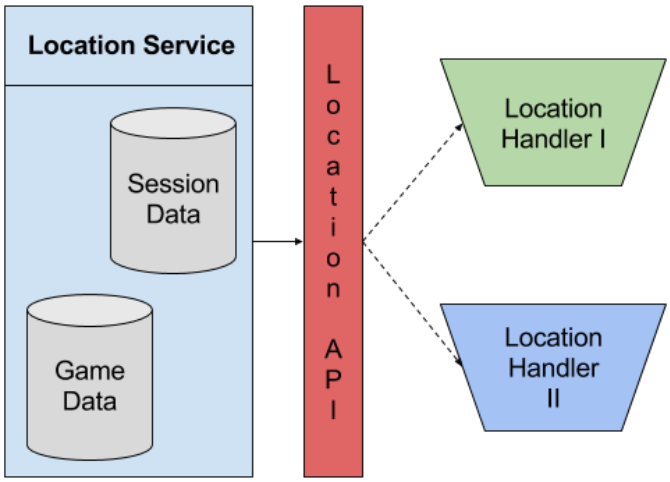


Figura 6.7: Exemplo de serviço de localização e exposição da API de localização para duas estratégias distintas.

Idealmente, seguindo esta arquitetura, tanto o servidor de jogo quanto o serviço de localização podem executar em ambientes com arquiteturas distintas. Nos testes descritos no Capítulo 5, por exemplo, uma mesma máquina executava estes componentes, que se comunicavam entre si e com os clientes móveis (Smartphones dos jogadores) através do middleware SDDL. Além disso, a API

poderia ser executada em uma outra arquitetura, seguindo o paradigma REST ou SOAP por exemplo, dado que os dados sejam de-serializados para arquivos XML ou JSON e aja uma interface para a API que segue o paradigma respectivo (GETs e POSTs para REST, por exemplo).

### 6.5.1.
### Fluxo de troca de mensagens

As chamadas de API realizadas pelos clientes móveis seguem o fluxo já descrito anteriormente: A informação da requisição chega no servidor de jogo, esse que repassa para o serviço de localização correspondente a sessão do jogador que enviou a requisição. O serviço de localização então identifica que Location Handler é responsável por este tipo de requisição e chama o método correspondente da API passando a informação de qual location handler deve ser usado, como mostrado na Figura 6.8.

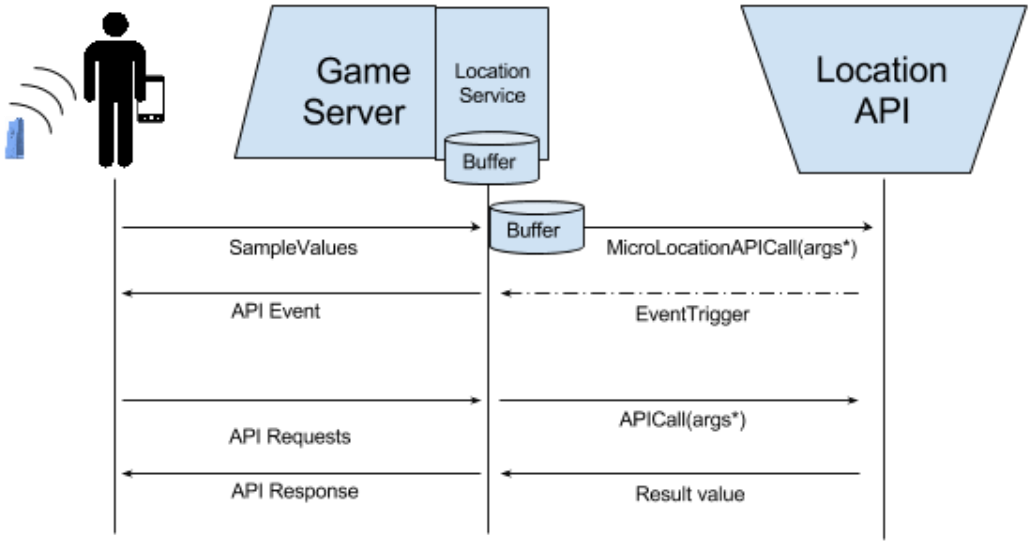


Figura 6.8: Fluxo para requisições e eventos da API de localização.

### 6.5.2.
### Responsabilidades de cada componente

A solução de microlocalização proposta é composta de três componentes distintos: serviços de localização, uma API de localização e um location handler para cada estratégia de localização. Como já descrito, o serviço de localização é um serviço que executa como um nó SDDL ou internamente a um servidor SDDL

(o servidor de jogo) e é capaz de receber pedidos de requisições vindas dos clientes móveis e transmitir engatilhamento de eventos (ambos através do servidor de jogo). O serviço de localização é vinculado a uma sessão de jogo e pode guardar em memória informações relevantes de localização para a sessão, como última posição conhecida dos jogadores e o buffer de amostras de RSSIs recebidos. O serviço chama a API, esta que tem acesso aos location handlers, um para cada estratégia de localização. A API em si pode estar em um endpoint na nuvem, em uma outra máquina na rede, executando na mesma máquina ou até mesmo interna ao servidor de jogo.

Considerando esse paradigma, foi determinado que:

- Os location handlers devem ser autocontidos, ou seja, capazes de responder às chamadas somente com os parâmetros de entrada e seus próprios métodos e constantes.
- A forma que a API é exposta independe do restante da aplicação, podendo ser através de requisições REST, data stream pelo SDDL (caso a API seja mais um nó SDDL) ou qualquer outra arquitetura. Isto vai depender unicamente de como a API é desenvolvida.
- O único componente que deveria guardar informações em memória para uso futuro é o próprio serviço de localização, podendo ter persistência através de um banco de dados, apesar de não ser necessário, já que ao término da sessão de jogo, o serviço pode parar de executar.
- O serviço deve ser capaz de transmitir dados para o servidor de jogo assincronamente, para que o fluxo do engatilhamento de eventos possa seguir até o cliente móvel.



### 6.5.3.
### Regras para definição de elementos da API

A API de localização proposta foi pensada de forma a ser expandida e modificada com novas requisições e eventos dependendo do jogo a ser desenvolvido. Dado que os serviços de localização e location handlers criados sigam as responsabilidades descritas para cada componente na subseção anterior, não há restrições para como devem ser criados elementos de API novos.

## 6.6.
## Análise da API

Para avaliar o uso da API, faremos duas análises distintas: abstrairemos a jogabilidade de alguns dos jogos pervasivos apresentados no Capítulo 3 dentro do contexto da API e, em seguida, descreveremos uma demo de um jogo fictício, *Area Control Game*, analisando sua possível implementação usando o middleware proposto e a API de Localização.

### 6.6.1
### Abstração de Jogos Pervasivos

- Human Pacman [38]

No jogo original, os jogadores Pacman tem que sobreviver a perseguição dos jogadores Fantasmas enquanto coletam frutas, estas que os tornam fortes temporariamente para vencerem os fantasmas. O jogo em si é como um pique-pega onde o pegador são os fantasmas até o momento em que um Pacman come uma fruta, passando ele então a poder pegar os fantasmas. Mapeando a área de jogo em múltiplas regiões mapeadas por beacons (ou talvez uma região grande), poderíamos identificar o "toque" de um jogador no outro através de um evento de *CoLocation*, identificando qual jogador pegou qual dependendo do papel e estado deles (Fantasma, Pacman normal e Pacman forte). As frutas poderiam ficar em pequenas regiões onde o evento de *IsInGrid* seria engatilhado quando um jogador entrasse nelas. Dessa forma, se um jogador for um Pacman normal, ele mudaria seu estado para Pacman forte.

-Hot Potato [39]

Neste jogo, os jogadores usam hardware especializado sensível a movimento para simular um jogo de batata quente, em que um timer faz a batata "explodir" na mão de um jogador caso ele demore muito para arremessa-la para outro jogador. Jogadores só podem arremessar batatas caso estejam próximos o suficiente de outros jogadores, e batatas são geradas para jogadores que saiam da área de jogo ao fugir de outros. A jogabilidade desse jogo pode ser abstraída através dos eventos de *IsInGrid*, para jogadores que saem da área de jogo serem punidos, e com requisições de *AproximateDistance,* síncronas (só permitindo arremessar a batata quando tem alguém próximo o suficiente) ou não (verificando a distância

no momento do input do usuário para arremessar a batata). Este jogo traz ainda a possibilidade de implementação de uma requisição do tipo *ListenForPlayers*, que verificaria a presença de todos os jogadores presentes em uma dada área em metros. Dentre os jogos listados como exemplo, este é o que o tempo de resposta é mais crucial para a jogabilidade, bem mais do que a precisão de localização em si. Isso o torna talvez o desafio mais interessante de uso da API e ajuste de janelas de busca de beacons em um trabalho futuro.

- Can You See Me Now? [54]

Um jogo como CYSMN, que mistura jogadores em um mundo virtual interagindo com jogadores no mundo real, também poderia ter sua jogabilidade abstraída, e até mesmo auxiliada, por microlocalização e a API proposta. Os jogadores do mundo virtual poderiam ter suas posições mapeadas para jogadores "fake" reais e suas atualizações de posição e buffer de sinais seriam realizadas da mesma forma, ou então poderia ocorrer uma modificação na requisição de *AproximateDistance* para que ela receba a posição (relativa ou absoluta) do jogador virtual para, então, calcular a distância para o jogador no mundo real, dado que já temos sua posição calculada por microlocalização. Além disso, poderiam haver novas formas de interação do mundo virtual para o real, onde os jogadores virtuais podem colocar armadilhas em células uma sub-região, notificando jogadores no mundo real quando os mesmos pisassem nelas.

- NetAttack [53]

NetAttack é um bom exemplo de exergame móvel dada a sua jogabilidade baseada em interação entre jogadores, mobilidade e ações condicionadas a proximidade entre jogadores. Nele, dois times diferentes têm que coletar artefatos (objetos físicos), estes que não podem ser coletados por um time caso jogadores de outro time estejam próximos. A área que um artefato se encontra pode ser uma pequena sub-região demarcada, de forma que os jogadores receberão notificação de um evento *IsInGrid* ou *IsInCell* ao entrarem nela. Além disso, caso jogadores estejam muito próximos (*ApproximateDistance* < um limite) ou até na mesma área ou célula (*CoLocation*), eles não podem coletar o artefato, que pode ser coletado talvez ao ficar tempo o suficiente no ponto físico em que ele se encontra (*TimeInCell*).

### 6.6.2.
### Análise teórica da demo: Area Control Game

Inicialmente planejado como uma demo a ser implementada com o Middleware proposto, *Area Control Game* é um jogo teórico de times que mescla o clássico jogo de pique-bandeira (*Capture the Flag*) com elementos e batalhas virtuais.

No jogo, que por ser uma partida já traz o conceito de sessão de jogo delimitada em espaço e tempo, jogadores se dividem em dois times com bases posicionadas como sub-regiões dentro de uma área de jogo. O objetivo do jogo é trazer todas as bandeiras da base inimiga para dentro da base do seu time. O time que primeiro for bem-sucedido nessa missão vence o jogo. As bandeiras, nesse exemplo, são objetos virtuais com posição pré-determinada no início do jogo e que podem ser movidas pelos jogadores caso não estejam sendo 'vigiadas' por um jogador do time adversário.

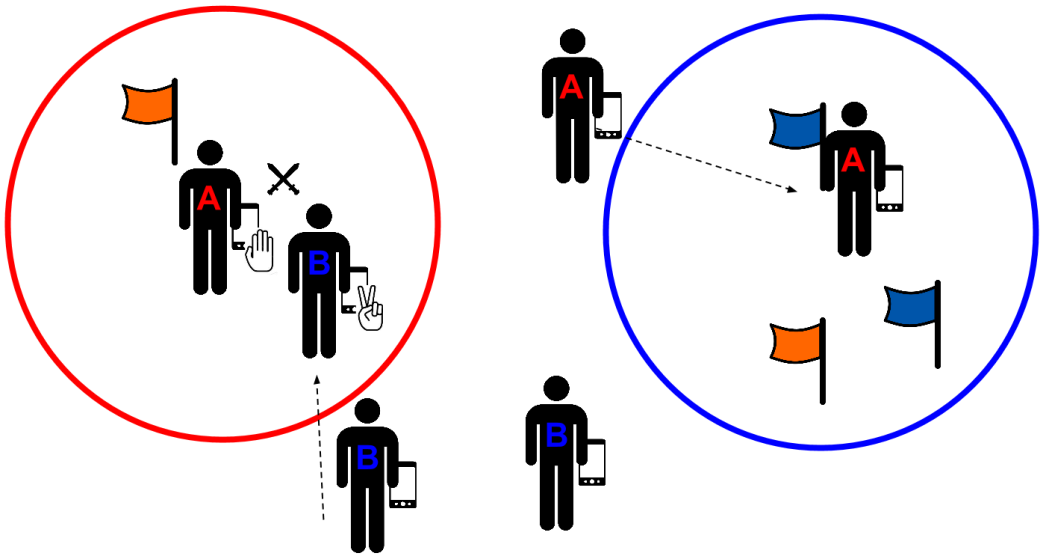


Figura 6.9: Um momento durante o jogo de Area Control Game

Caso haja um jogador de outro time próximo a bandeira virtual que um jogador tente capturar, os jogadores entram em uma batalha virtual, abstraída para um jogo de pedra, papel e tesouras para simplificar. O jogador que perder não consegue "ver" bandeiras nem jogadores durante um período de tempo, não podendo proteger outras bandeiras durante o período. Os jogadores podem entrar

em batalhas a qualquer momento, mesmo sem bandeiras próximas, desde que estejam próximos o suficiente. O jogo termina quando um dos times capturar todas as bandeiras do adversário.

Em um jogo como este, há desafios de orquestração, localização e execução bem claros:

- Antes do jogo iniciar, a área de jogo dever ser preparada, as sub-regiões mapeadas e as bandeiras virtuais com suas posições definidas;

- Devemos conseguir a localização dos jogadores em relação um ao outro e em relação com as bandeiras a qualquer momento e em um curto espaço de tempo, mesmo com os jogadores correndo. Fazendo um paralelo com o pique-bandeira no "mundo real", a batalha deveria ocorrer em uma distância de toque entre dois jogadores, menor que um metro;

- Devemos capturar as ações dos jogadores, seja para batalhar com outros jogadores ou para mover uma bandeira de lugar;

- Os jogadores devem ser capazes de entender os acontecimentos do jogo sem precisar olhar para o que está acontecendo no "mundo real", apenas pelo Smartphone.

Considerando um jogo desse tipo, fica claro o benefício que o uso da API de Localização em conjunto com o middleware teria para sua implementação e execução plena. Enquanto o middleware lidaria com a criação e gerência de sessões de jogo, assim como a orquestração inicial do mesmo (em conjunto com a demarcação física da área usando o Beacons BLE), usaríamos os seguintes componentes da API de localização para possibilitar o uso de localização com a precisão desejada:

- Configuração de tempo de resposta: Janela de busca de beacons de 100ms com intervalo de 300ms por envio de amostra garantindo aproximadamente 500ms para uma atualização de localização;

- Interface de métodos da API disponíveis para o Smartphone com métodos de *IsInGrid* e *ListenForPlayers*;

- Temos a verificação de proximidade dada através do raio das sub-regiões: Com a requisição de *IsInGrid*, abstraímos a "visão" do jogador para enxergar o que está no grid com ele, ou seja, seu raio de visão passa a ser o raio da sub-região que ele se encontra;

- Já para a as ações de batalha ou captura de bandeira, podemos usar a requisição de *ListenForPlayers*, que pode verificar em um raio de ação especifico a proximidade de jogadores ou de outros elementos do jogo, no caso, as bandeiras. A ação a ser tomada com o elemento, captura de bandeira ou batalha com jogador de um outro time, pode ser tomada somente se o jogador estiver próximo o suficiente;

- Com isso, há ainda uma requisição que pode ser feita pelo servidor, mas deixada escondida do jogador, um *AproximateDistance* para os jogadores do outro time na sub-região quando um jogador tentar capturar uma bandeira. Se o jogador do outro time estiver próximo o suficiente, o jogo pode avisa-lo que ele pode defender essa bandeira, ou até mesmo obriga-lo a defender essa bandeira, dependendo de como a regra for estabelecida;

- O uso de um location handler que implemente uma estratégia de localização suficientemente precisa para as requisições desse jogo, mas que não impacte o tempo de resposta com seu processamento.

Dessa forma, com esses elementos da API de Localização dispostos para o jogo e com o servidor lidando com as regras de condição de vitória, ao rastrear a última posição das bandeiras virtuais, os requisitos para execução do jogo tornam-se satisfeitos.

## 6.7. Discussão

A API de localização proposta tem como objetivo auxiliar o desenvolvimento de exergames móveis, dado que jogos dessa categoria tipicamente requerem algum tipo de solução para microlocalização. Através da arquitetura proposta, é possível expandir a API de localização criando novas requisições e eventos, criar outros location handlers no qual a solução ou estratégia de microlocalização seja diferente da proposta ou até mesmo criar novos serviços e APIs especializadas que abstraiam outros tipos de funcionalidades para exergames móveis.

Apesar disso, há problemas de arquitetura e implementação que podem ser endereçados em trabalhos futuros. A arquitetura que lida com serviços vinculados a sessões pode não escalar bem dependendo do tipo de jogo, caso tenha muitas

sessões, o que não foi devidamente testado. Também não é confiável afirmar que a arquitetura é válida dado que somente um location handler foi desenvolvido para ela.

Sobre a API em si, a avaliação da mesma ainda carece de validez, dado que é apenas um exercício de abstração e exemplificação de jogos já existentes. Seria interessante que houvesse a implementação de dois ou três demos de exergames móveis, entre eles o Area Control Game, que exemplificassem na prática o uso de requisições e eventos da API e sua integração com o restante da arquitetura.

Uma linha de trabalho futura consiste na possibilidade de eventos vindos dos jogadores, como movimentos do celular e inputs manuais (botões no celular ou tela, por exemplo) e sua interpretação pelo servidor de jogo, e analisar se faz sentido incluir esse tipo de tratamento de eventos na API ou se podem ser resolvidos externamente.

Uma outra possível melhoria seria incluir um cache local no cliente móvel para determinadas consultas de localização, especialmente se são de informações que não mudam durante o jogo, como o mapa de rádio, por exemplo. Essas funcionalidades e melhorias contribuiriam para a arquitetura proposta, no futuro, ser apresentável como um middleware para exergames móveis.

# 7
# Conclusão e Trabalhos Futuros

Neste trabalho, procuramos, antes de mais nada, definir o contexto cultural e tecnológico no qual nos fundamentamos. Nossa motivação do contexto tecnológico no qual tecnologias de sensoriamento e IoT estão cada vez mais acessíveis, ao mesmo tempo que jogos baseados em localização, em sua maioria usando apenas GPS, estão ficando mais comuns. A partir disso, vimos que tem toda uma área de diferentes categorias de jogos pervasivos com requisitos tecnológicos distintos, e por isso não explorados comercialmente, dentre eles exergames móveis.

Dessa forma, o foco dos primeiros capítulos foi pavimentar bem o conceito de exergame móvel, referenciando trabalhos que também buscam categorizar jogos pervasivos de modo a nos levar a uma visão sólida dos desafios relativos a implementação desses tipos de jogo, focando nos desafios de localização em tempo real dos jogadores. Nos capítulos seguintes, apresentamos o middleware criado e a estratégia de microlocalização de prova de conceito usando beacons BLE para, então, apresentar a contribuição principal desse trabalho, a API de localização com requisições e eventos. Este trabalho visa contribuir não somente para o desenvolvedor de jogos pervasivos, como também para outros desenvolvedores de aplicações baseadas em localização, proximidade e sensoriamento no futuro.



## 7.1.
## Principais contribuições

Podemos listar como principais contribuições desse trabalho, incialmente, a análise de diferentes categorias de jogos pervasivos seguida de uma definição considerando contexto tecnológico e cultural de exergames móveis. Como categorização e análise de jogos pervasivos até hoje ainda é um tema muito discutido, achamos válido tomar o tempo necessário para solidificar o conceito, de modo a usá-lo no restante do trabalho. Essa análise e definição pode contribuir

para futuras pesquisas na área, dado que existem poucos trabalhos que lidam especificamente com exergames móveis.

Os seguintes temas abordados tinham como objetivo criar uma base para a apresentação da contribuição principal. Primeiro, o middleware para exergames móveis, que tem como objetivo capacitar o desenvolvimento de exergames móveis que lidam com microlocalização, com a criação de sessões de jogos, gerenciamento de jogadores, área de jogo e sub-regiões, além de vincular beacons a sub-regiões mapeáveis e incluir serviços especializados que usem APIs para resolver funcionalidades especificas do jogo, como microlocalização. A arquitetura proposta para esse middleware pode ainda ser melhor definida, de forma a possibilitar a criação de diferentes jogos pervasivos que não somente exergames móveis, além da possibilidade de uma futura expansão para um Framework de Exergames Móveis.

Em seguida, a análise da estratégia de fingerprinting com algoritmo determinístico para comparação de amostras, em conjunto com o uso de beacons BLE para mapear as sub-regiões de microlocalização. Realizamos testes para averiguar a possibilidade de usar esta técnica em conjunto com essa tecnologia, estas pouco exploradas em ambientes outdoor e em jogos pervasivos, muitas vezes requerendo a resposta em tempo real. Apesar dos resultados não conclusivos para afirmar que a estratégia funciona, vimos através de diversos trabalhos que a estratégia e execução podem ser aprimoradas em vários aspectos para conseguirmos uma precisão melhor, além da possibilidade de usarmos métodos probabilísticos e aumentarmos ainda mais a precisão.

Finalmente, a API de localização proposta, dentro do paradigma de um serviço de localização que se comunica com a API exposta para a execução de diferentes estratégias de localização através de location handlers internos a ela. Apresentamos diferentes requisições e eventos que possibilitam, através deles, a construção de diferentes jogabilidades para exergames móveis e até outros tipos de jogos pervasivos e baseados em localização. Achamos que evoluir a API de localização é um bom caminho para melhorar a proposta, porém, o ideal seria pensar na arquitetura como um todo, propondo uma melhor integração entre os serviços especializados, o servidor de jogo e os clientes móveis carregados pelos jogadores.

## 7.2.
## Trabalhos Futuros

Por este trabalho lidar com diferentes conceitos e tecnologias, há muitas melhorias futuras e linhas de pesquisa que podem seguir a partir dele. Muitas dessas linhas já foram discutidas na sessão de discussão dos respectivos capítulos, porém algumas serão apresentas a seguir.

- *API de localização para diferentes contextos de aplicação*

Apesar da motivação e tema geral do trabalho advir de jogos pervasivos, uma forma evoluída da API de localização poderia auxiliar muito o desenvolvimento de aplicações em área como cidades inteligentes, casas inteligentes, medicina, acessibilidade, urbanismo e até arte. A implementação de aplicações nessa área em conjunto com a evolução da API e testes com diferentes estratégias e tecnologias de beacons é importante para a evolução deste trabalho para além do contexto de jogos pervasivos.

- *Implementação de jogos demo – Area Control Game*

Certamente uma das maiores faltas desse trabalho foi implementação de jogos demo para testar tanto a API de localização como o middleware. A implementação de jogos que usem diferentes jogabilidade, configurações de times e talvez tecnologias de sensores e beacons é essencial para que se dê continuidade a este trabalho. Um grande candidato à primeiro demo de prova de conceito é o próprio Area Control Game descrito no Capítulo 6.

- *Expansão dos testes*

Há a necessidade de realização dos testes da estratégia de localização em mais ambientes com outras características, diferentes números de jogadores, outros algoritmos de comparação de dados (determinísticos e probabilísticos), além de testes de campo com vários jogadores em possíveis demos de jogos simples. Também deve-se entender melhor o impacto de configurações e condições especificas de hardware (como a bateria e janela de busca para os beacons BLE) na precisão atingida nos testes.

- *Expandir o middleware para outras tecnologias*

O middleware proposto deve prever e ser capaz de lidar com componentes de outras tecnologias que não sejam somente as usadas neste trabalho. Criar um serviço separado do servidor de jogo que consuma de uma API de localização

REST em conjunto com a implementação de aplicativos de jogos para outros sistemas móveis, como iOS e Windows Phone, pode ser um caminho positivo.

A área de jogos pervasivos tem muito a evoluir para que se torne um gênero competitivo no mercado de jogos, assim como a aplicações IoT ainda tem um longo caminho para que sejam realmente pervasivas e contribuintes no dia a dia da sociedade. Esperamos que este trabalho contribua com esse objetivo e que inspire outros, mesmo que pouco.

# 8

# Referências Bibliográficas


1 Burrus, Daniel, “**The Internet of Things Is Far Bigger Than Anyone Realizes - Wired**” 2014. [Online] Disponível: http://www.wired.com/insights/2014/11/the-internet-of-things-bigger/. [Acessado: 29-11-2015]

2 McKendrick, Joe, “**The Internet Of Things Dominates Recent Mergers & Acquisitions - Forbes**” 2015. [Online] Disponível: http://www.forbes.com/sites/joemckendrick/2015/11/28/the-internet-of-things-dominates-recent-mergers-acquisitions/. [Acessado: 29-11-2015]

3 Intel, “**The Internet of Things starts with Intel Inside - Intel**” [Online] Disponível: http://www.intel.com/content/www/us/en/internet-of-things/overview.html. [Acessado: 29-11-2015]

4 Cisco, “**Internet of Things (IoT) - Cisco**” [Online] Disponível: http://www.cisco.com/web/solutions/trends/iot/overview.html. [Acessado: 29-11-2015]

5 Lea, Nick [Nortec] & McMullin, Rick [bitHeads], “**Do end users care about the Internet of Things? - bitHeads**” [Online] Disponível: http://www.bitheads.com/do-end-users-care-about-the-internet-of-things/. [Acessado: 29-11-2015]

6 Page, Barnaby, “**End users more interested in IoT than developers - ScreenMediaMag**” [Online] Disponível: http://www.screenmediamag.com/news-list/4677-end-users-more-interested-in-iot-than-developers. [Acessado: 29-11-2015]

7 **Gartner Group**. [Online] Disponível: http://blogs.gartner.com/earl-perkins/2014/03/05/getting-past-the-word-games-to-secure-the-internet-of-things/ [Acessado: 30-11-2015]

8 Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "**The internet of things: A survey.**" Computer networks 54.15 (2010): 2787-2805.

9 **Pew Research Center**. [Online] Disponível: http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/ [Acessado: 30-11-2015]

10 Nieuwdorp, Eva. "**The pervasive discourse: an analysis.**" Computers in Entertainment (CIE) 5.2 (2007): 13.

11 Benford, Steve, Carsten Magerkurth, and Peter Ljungstrand. "**Bridging the physical and digital in pervasive gaming."** Communications of the ACM 48.3 (2005): 54-57.

12 Broll, Wolfgang, et al. "**Meeting technology challenges of pervasive augmented reality games.**" Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games. ACM, 2006.

13 Walther, Bo Kampmann. "**Atomic actions--molecular experience: theory of pervasive gaming**." Computers in Entertainment (CIE) 3.3 (2005): 4-4.

14 Magerkurth, Carsten, et al. "**Pervasive games: bringing computer entertainment back to the real world.**" Computers in Entertainment (CIE) 3.3 (2005): 4-4.

15 Wylie, Carlos Garcia, and Paul Coulton. "**Mobile exergaming.**" Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology. ACM, 2008.
16 Chittaro, Luca, and Riccardo Sioni. "**Turning the classic snake mobile game into a location–based exergame that encourages walking**." Persuasive Technology. Design for Health and Safety. Springer Berlin Heidelberg, 2012. 43-54.
17 Macvean, Andrew, and Judy Robertson. "**iFitQuest: a school based study of a mobile location-aware exergame for adolescents.**" Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services. ACM, 2012.
18 Görgü, Levent, et al. "**Freegaming: Mobile, collaborative, adaptive and augmented exergaming.**" Mobile Information Systems 8.4 (2012): 287-301.
19 **Google Ingress**. [Online] Disponível: https://www.ingress.com/ [Acessado 01-12-2015]
20 **Zombies Run!**. [Online] Disponível: https://zombiesrungame.com/ [Acessado 01-12-2015]
21 **Geocatching.** [Online] Disponível: https://www.geocaching.com/ [Acessado 01-12-2015]
22 McNamara, Joel (2008). **GPS For Dummies**. John Wiley & Sons. p. 59. ISBN 0-470-45785-6
23 **US-Government**. [Online] Disponível: http://www.gps.gov/systems/gps/performance/accuracy/ [Acessado 02-12-2015]
24 **Bluetooth**. [Online] Disponível: http://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy [Acessado 03-12-2015]
25 **Google Android**. [Online] Disponível: https://www.android.com/ [Acessado 03-12-2015]
26 **Apple**. [Online] Disponível: http://www.apple.com/ios/ [Acessado 03-12-2015]
27 **Arduino**. [Online] Disponível: https://www.arduino.cc/ [Acessado 03-12-2015]
28 **RaspberryPi**. [Online] Disponível: https://www.raspberrypi.org/ [Acessado 03-12-2015]
29 **Arm**. [Online] Disponível: https://www.arm.com/markets/internet-of-things-iot.php [Acessado 03-12-2015]
30 **Techcrunch**. [Online] Disponível: http://techcrunch.com/2015/03/10/cheaper-sensors-will-fuel-the-age-of-smart-everything/ [Acessado 03-12-2015]
31 Rashid, O., Mullins, I., Coulton, P., & Edwards, R. (2006). **Extending Cyberspace: Location Based Games Using Cellular Phones**. Computers in Entertainment, 4(1), 4. doi:10.1145/1111293.1111302
32 Geddes, O. (2014). **A Guide to Bluetooth Beacons**, (September). Retrieved from http://www.gsma.com/digitalcommerce/wp-content/uploads/2013/10/A-guide-to-BLE-beacons-FINAL-18-Sept-14.pdf
33 Stojanović, D. H., & Stojanović, N. M. (2014). **Indoor Localization and Tracking: Methods, Technologies and Research Challenges**. *Facta*

*Universitatis, Series: Automatic Control and Robotics*, *13*(Iii 43007), 57–72.

34 Obón, D. I. P. T., Alk, T. I. H. F., & Aier, M. A. M. (2013). **Context Awareness in WBANs: A Survey on Medical and Non-Medical Applications**, (August), 30–37.

35 Sun, G., Chen, J., Guo, W., & Liu, K. J. R. (2005). **Signal processing techniques in network-aided positioning: A survey of state-of-the-art positioning designs**. *IEEE Signal Processing Magazine*, *22*(4), 12–23. doi:10.1109/MSP.2005.1458273

36 Bulusu, Nirupama, et al. "**Scalable coordination for wireless sensor networks: self-configuring localization systems**." International Symposium on Communication Theory and Applications (ISCTA 2001), Ambleside, UK. 2001.

37 Leonidis, Asterios, et al. "**An Intelligent Hotel Room**." *Ambient Intelligence*. Springer International Publishing, 2013. 241-246.

38 Cheok, A. D., Goh, K. H., Liu, W., Farbiz, F., Fong, S. W., Teo, S. L., … Yang, X. (2004). **Human Pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing**. *Personal and Ubiquitous Computing*, *8*(2), 71–81. doi:10.1007/s00779-004-0267-x

39 Akribopoulos, O., Logaras, M., Vasilakis, N., Kokkinos, P., Mylonas, G., Chatzigiannakis, I., & Spirakis, P. (2009). **Developing multiplayer pervasive games and networked interactive installations using ad hoc mobile sensor nets**. In *Proceedings of the International Conference on Advances in Computer Enterntainment Technology - ACE '09* (p. 174). New York, New York, USA: ACM Press. doi:10.1145/1690388.1690418

40 Montola, M., Stenros, J., & Waern, A. (2009). **Pervasive games: theory and design.** Morgan Kaufmann Publishers Inc..

41 Montola, Markus. "**Exploring the edge of the magic circle: Defining pervasive games**." Proceedings of DAC. Vol. 1966. 2005.

42 Sinclair, Jeff, Philip Hingston, and Martin Masek. **"Considerations for the design of exergames**." Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia. ACM, 2007.

43 **Nintendo Wii Fit.** [Online] Disponível: http://wiifit.com/ [Acessado 04-12-2015]

44 **Ubisoft Just Dance**. [Online] Disponível: http://just-dance.ubi.com/ [Acessado 04-12-2015]

45 **Nintendo Wii Mote**. [Online] Disponível: http://www.nintendo.com/wiiu/accessories [Acessado 04-12-2015]

46 **Microsoft Kinect**. [Online] Disponível: https://dev.windows.com/en-us/kinect [Acessado 04-12-2015]

47 **Sony Playstation Move**. [Online] Disponível: http://playstationmove.com/ [Acessado 04-12-2015]

48 Laine, Teemu H., and Carolina Islas Sedano. "**Distributed Pervasive Worlds: The Case of Exergames.**" Journal of Educational Technology & Society 18.1 (2015): 50-66.

49 Herbst, I., Braun, A.-K., McCall, R., & Broll, W. (2008). **TimeWarp: interactive time travel with mobile mixed reality game**. In:

Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI), 235–244. doi:10.1145/1409240.1409266

50 Lankoski, P., Heliö, S., Nummela, J., Lahti, J., Mäyrä, F., & Ermi, L. (2004). **A Case Study in Pervasive Game Design: The Songs of North.** Proceedings of the Third Nordic Conference on Human-Computer Interaction - NordiCHI '04, 413–416. doi:10.1145/1028014.1028083

51 Carrigy, T., Naliuka, K., Paterson, N., & Haahr, M. (2010). **Design and evaluation of player experience of a location-based mobile game**. NordiCHI 2010: Extending Boundaries - Proceedings of the 6th Nordic Conference on Human-Computer Interaction, 92–101. doi:10.1145/1868914.1868929

52 Benford, Steve, et al. "**Uncle Roy All Around You: Implicating the city in a location-based performance.**" Proc. Advances in Computer Entertainment (ACE 2004) 21 (2004): 47.

53 Lindt, Irma, and Wolfgang Broll. "**NetAttack–First Steps Towards Pervasive Gaming**." ERCIM NEWS Special Issue on Games Technology 57 (2004): 49-50.

54 Benford, S., Crabtree, A., Flintham, M., Drozd, A., Anastasi, R., Paxton, M., … Row-Farr, J. (2006). **Can you see me now?** ACM Transactions on Computer-Human Interaction, 13(1), 100–133. doi:10.1145/1143518.1143522

55 Sotamaa, Olli. "**All The World's A Botfighter Stage: Notes on Location-based Multi-User Gaming.**" CGDC Conf.. 2002.

56 **Market Search World**. [Online] Disponível: http://www.marketresearchworld.net/content/view/5415/76/ [Acessado: 08-12-2015]

57 **It's Alive!.** [Online] Disponível: http://web.archive.org/web/20050207084809/http://www.itsalive.com/page.asp [Acessado: 08-12-2015]

58 Benford, S., Ramsden, A., & Roussos, G. (2005). **Future Location-Based Experiences**, 1–17.

59 Ohlenburg, J., Herbst, I., Lindt, I., Fröhlich, T., & Broll, W. (2004). **The MORGAN framework: enabling dynamic multi-user AR and VR projects.** *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 166–169. doi:http://doi.acm.org/10.1145/1077534.1077568

60 Valente, Luis, Bruno Feijo, and Julio Cesar Sampaio do Prado Leite. "**Mapping Quality Requirements for Pervasive Mobile Games**." Requirements Engineering, 2015. doi:10.1007/s00766-015-0238-y.

61 Tutzschke, J.-P., & Zukunft, O. (2009). **Frap: A Framework for Pervasive Games**. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems - EICS '09* (p. 133). New York, New York, USA: ACM Press. doi:10.1145/1570433.1570459

62 **Agent Factory**. [Online] Disponível: http://www.agentfactory.com/index.php/Main_Page [Acessado: 11-12-2015]

63 David, L., Vasconcelos, R., Alves, L., André, R., & Endler, M. (2013). **A DDS-based middleware for scalable tracking, communication and**

**collaboration of mobile nodes**. *Journal of Internet Services and Applications*, *4*(1), 1-15. DOI=10.1186/1869-0238-4-16

64 Endler M., Baptista G., David L., Vasconcelos R., Malcher M., Pantoja V., Pinheiro V., Viterbo J. (2011). **ContextNet: context reasoning and sharing middleware for large-scale pervasive collaboration and social networking**. In *Proceedings of the Workshop on Posters and Demos Track (PDT '11). ACM, New York, NY, USA,* Article 2, 2 pages. DOI=10.1145/2088960.2088962

65 Silva, L.D., Roriz, M., Endler, M., (2013). **MR-UDP: Yet another reliable user datagram protocol, now for mobile nodes**, Monografias em Ciência da Computação, nr. MCC 06/13, Departamento de Informática, PUC-Rio, ISSN 0103-9741

66 Talavera, L.E., Endler, M., Vasconcelos, I., Vasconcelos, R.,Cunha, M., Silva, F.S., (2014). **The Mobile Hub Concept: Enabling applications for the Internet of Mobile Things**, Monografias em Ciência da Computação, nr. MCC 04/14, Departamento de Informática, PUC-Rio, ISSN 0103-9741

67 Bulusu, N., Heidemann, J., & Estrin, D. (2000). **GPS-less low-cost outdoor localization for very small devices**. *IEEE Personal Communications*, *7*(5), 28–34. doi:10.1109/98.878533

68 Chu, H.-C., & Jan, R.-H. (2007). **A GPS-less, outdoor, self-positioning method for wireless sensor networks**. *Ad Hoc Networks*, *5*(5), 547–557. doi:10.1016/j.adhoc.2006.03.004

69 Boukerche, a, Oliveira, H. a B., Nakamura, E. F., & Loureiro, a a F. (2007). **Localization systems for wireless sensor networks**. *IEEE Wireless …*, *14*(December), 6–12. doi:10.1109/MWC.2007.4407221

70 Reichenbach, F., & Timmermann, D. (2006). **Indoor localization with Low complexity in wireless sensor networks**. *Proc. IEEE Int. Conf. Ind. Informatics*, *00*, 1018–1023. doi:10.1109/INDIN.2006.275737

71 Alhmiedat, T., & Samara, G. (2013). **An Indoor Fingerprinting Localization Approach for ZigBee Wireless Sensor Networks**, *105*(2), 190–202.

72 Faragher, R., & Harle, R. (2015). **Location Fingerprinting with Bluetooth Low Energy Beacons**. *IEEE Journal on Selected Areas in Communications*, *PP*(99), 1–1. doi:10.1109/JSAC.2015.2430281

73 Zhao, X., Xiao, Z., Markham, A., Trigoni, N., & Ren, Y. (2014). **Does BTLE measure up against WiFi? A comparison of indoor location performance**. 20th European Wireless Conference, 1–6.

74 Martin, E., Vinyals, O., Friedland, G., & Bajcsy, R. (2010). **Precise indoor localization using smart phones**. In *Proceedings of the international conference on Multimedia - MM '10* (p. 787). New York, New York, USA: ACM Press. doi:10.1145/1873951.1874078

75 Stojanović, D. H., & Stojanović, N. M. (2014). **Indoor Localization and Tracking: Methods, Technologies and Research Challenges**. *Facta Universitatis, Series: Automatic Control and Robotics*, *13*(Iii 43007), 57–72.

76 Lee, Y., Ju, Y., Min, C., Yu, J., & Song, J. (2012). **MobiCon: Mobile context monitoring platform: Incorporating context-awareness to smartphone-centric personal sensor networks**. *Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc*

*Communications and Networks Workshops*, *1*, 109–111. doi:10.1109/SECON.2012.6275765

77 Enterprise, D. (2014). **API-fication: Core building block of the digital enterprise**, (August).

78 Hwang, I., Lee, Y., Park, T., & Song, J. (2012). **Toward a mobile platform for pervasive games**. In *Proceedings of the first ACM international workshop on Mobile gaming - MobileGames '12* (p. 19). New York, New York, USA: ACM Press. doi:10.1145/2342480.2342486

79 Küpper, A. (2005). **Location-based services: fundamentals and operation**. John Wiley & Sons.

80 Reiter, Gil. "**Wireless connectivity for the Internet of Things**." Europe 433 (2014): 868MHz.

81 Honkavirta, V., Perälä, T., Ali-Löytty, S., & Piché, R. (2009). **A comparative survey of WLAN location fingerprinting methods**. Proceedings - 6th Workshop on Positioning, Navigation and Communication, WPNC 2009, 2009, 243–251. http://doi.org/10.1109/WPNC.2009.4907834

82 **Texas Instruments CC2541 SensorTag**: http://processors.wiki.ti.com/index.php/CC2541_SensorTag [Acessado: 14-01-2016]

83 Saxena, M., Gupta, P., & Jain, B. N. (2008). **Experimental analysis of RSSI-based location estimation in wireless sensor networks**. 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08), 503–510. http://doi.org/10.1109/COMSWA.2008.4554465

84 Geddes, O. (2014). **A Guide to Bluetooth Beacons**, (September). Retrieved from http://www.gsma.com/digitalcommerce/wp-content/uploads/2013/10/A-guide-to-BLE-beacons-FINAL-18-Sept-14.pdf

85 Kril, Paul; InfoWorld: "**Make way for SOA 2.0**". [Online] Disponível: http://www.infoworld.com/article/2654672/applications/make-way-for-soa-2-0.html [Acessado: 17-01-2016]

86 Zahradnik, F. **GPS About** (2014). [Online] Disponível: http://gps.about.com/od/glossary/g/A-GPS.htm [Acessado: 20/01/2016]

87 Florian Mueller, Stefan Agamanolis, and Rosalind Picard. 2003. **Exertion interfaces: sports over a distance for social bonding and fun**. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03). ACM, New York, NY, USA, 561-568. DOI=http://dx.doi.org/10.1145/642611.642709

88 Mueller, F. `Floyd', Stevens, G., Thorogood, A., O'Brien, S., & Wulf, V. (2007). **Sports over a Distance.** Personal and Ubiquitous Computing, 11(8), 633–645. http://doi.org/10.1007/s00779-006-0133-0

89 Mueller, F., Agamanolis, S., Gibbs, M., Vetere, F.: Remote Impact: **Shadowboxing over a Distance**. In: Extended Abstracts of CHI'08, pp. 2291--2296. ACM, Florence (2008).

90 Mueller, F., Gibbs, M., Vetere, F., Agamanolis, S.: **Design Space of Networked Exertion Games Demonstrated by a Three-way Physical Game Based on Table Tennis**. ACM Computers in Entertainment, 6, 1--31, (2008).