UNIVERSIDADE FEDERAL FLUMINENSE

PERON REZENDE DE SOUSA

PROJETO E IMPLEMENTAÇÃO DE UMA ARQUITETURA ESCALÁVEL DE MOBILE CROWDSOURCING

NITERÓI

2019

UNIVERSIDADE FEDERAL FLUMINENSE

PERON REZENDE DE SOUSA

PROJETO E IMPLEMENTAÇÃO DE UMA ARQUITETURA ESCALÁVEL DE MOBILE CROWDSOURCING

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito para a obtenção do Grau de Doutor em Computação. Área de concentração: SISTEMAS DE COMPUTAÇÃO.

Orientador:
ANTONIO AUGUSTO DE ARAGÃO ROCHA

Co-orientador:
MARCOS DE OLIVEIRA LAGE FERREIRA

NITERÓI

2019

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S725p Sousa, Peron Rezende de
Projeto e implementação de uma arquitetura escalável de mobile crowdsourcing / Peron Rezende de Sousa ; Antonio Augusto de Aragão Rocha, orientador ; Marcos de Oliveira Lage Ferreira, coorientador. Niterói, 2019.
130 f. : il.

Tese (doutorado)-Universidade Federal Fluminense, Niterói, 2019.

DOI: http://dx.doi.org/10.22409/PGC.2019.d.07314896755

1. Arquitetura de software. 2. Sistemas distribuídos em tempo real. 3. Geolocalização. 4. Marketing de rede. 5. Produção intelectual. I. Rocha, Antonio Augusto de Aragão, orientador. II. Ferreira, Marcos de Oliveira Lage, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

PERON REZENDE DE SOUSA

PROJETO E IMPLEMENTAÇÃO DE UMA ARQUITETURA ESCALÁVEL DE MOBILE CROWDSOURCING

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: SISTEMAS DE COMPUTAÇÃO.

Aprovada em Junho de 2019.

BANCA EXAMINADORA

Prof. Antonio Augusto de Aragão Rocha - Orientador, UFF

Prof. Célio Vinicius Neves de Albuquerque, UFF

Prof. Leandro Augusto Frata Fernandes, UFF

Prof. Pedro Braconnot Velloso, UFRJ

Prof. José Ferreira de Rezende, UFRJ

NITERÓI

2019

Dedico esse trabalho à minha adorável filha Alícia que nasceu durante essa trajetória e em memória da minha tia Zoraida.

Agradecimentos

Agradeço a Deus por me proteger em todos os meus caminhos e a todos que direta ou indiretamente me ajudaram durante essa jornada, em especial ao meu orientador Antonio Rocha (Guto) e ao meu co-orientador Marcos Lage, agradeço a ambos pela compreensão e tempo dedicado nos finais de semana e altas horas da noite. A paciência e bom humor dos professores Célio Albuquerque, José Viterbo, Helena Cristina, Fabio Protti, Igor Moraes e Celso Ribeiro. A secretaria do instituto de computação, em especial, Hélio Augusto, Teresa Cancela e Vanessa Braganholo. Aos colegas de pós-graduação Gabriel Almeida de Oliveira, Juan Vieira, Rhodrigo Santana, Diogo Robaina e Alexandre Luiz. A Vitor Farias pela ajuda com o FIBRE e Iara Machado, da RNP, por todo suporte com o PlanetLab. A Aditya Yadav e David Westbrook da UMass pelo suporte com MSocket. Agradeço também aos integrantes da comissão examinadora, pois suas observações, dicas e críticas ajudaram a melhorar a versão final deste trabalho, ainda que a falta de tempo não tenha permitido aproveitar todas as orientações.

Resumo

A popularização dos dispositivos móveis com múltiplos recursos, a localização em tempo real e a disponibilidade da "computação humana", provida por uma multidão global de usuários, motivaram o surgimento dos sistemas de Mobile Crowdsourcing. Esse novo paradigma apresenta vários requisitos, como assertividade na atribuição de tarefas, segurança, mecanismo de incentivo, suporte à mobilidade e escalabilidade. Nesta tese, escalabilidade é a capacidade de atender a uma demanda crescente sem perda de performance, ponto ignorado em muitas propostas que fazem uso do modelo tradicional (cliente-servidor). A solução deste problema também é o foco principal deste trabalho. Para resolver a questão revisamos o estado da arte e analisamos onde cada proposta falhou. Idealizamos uma nova arquitetura, intitulada Snow Flurry, e trabalhamos na implementação da mesma para realização de uma prova de conceito que confirmou sua viabilidade. Experimentos avaliaram o desempenho dos componentes que integram a solução e os resultados mostraram que a arquitetura oferece uma redução na latência entre 26,31% e 32,03%. Outro ponto interessante é que a Snow Flurry, ao contrário do estado da arte, não apresenta sobrecargas, alto custo e a necessidade de gerenciar dezenas de máquinas espalhadas pelo mundo. Ela também foi a primeira com suporte à mobilidade para sistemas de Mobile Crowdsourcing. Adicionalmente, esta tese contribui com a implementação do MktPoints que é um novo mecanismo de incentivo atrelado a um serviço de reputação. Inspirado em marketing de rede e nos efeitos psicológicos das recompensas variáveis, o MktPoints não compromete a escalabilidade do sistema e atua na atração/manutenção de trabalhadores sem necessariamente envolver motivadores extrínsecos.

Palavras-chave: Crowdsourcing, Internet do Futuro, Escalabilidade.

Abstract

The popularization of mobile devices with multiple resources, the real-time location, and the availability of "human computing", provided by a global crowd of users, motivated the emergence of Mobile Crowdsourcing systems. This new paradigm presents several requirements such as assertiveness in task assignment, security, incentive mechanism, mobility support and scalability. In this thesis, scalability is the ability to meet increasing demand without performance loss, a point ignored in many proposals that make use of the traditional model (client-server). The solution of this problem is also the main focus of this work. To address the issue we review the state of the art and analyze where each proposal failed. We designed a new architecture, called Snow Flurry, and worked on its implementation to carry out a proof of concept that confirmed its feasibility. Experiments evaluated the performance of the components that make up the solution and the results showed that the architecture offers a reduction in latency between 26.31% and 32.03%. Another interesting point is that Snow Flurry, unlike the state of the art, has no overhead, no cost and no need to manage dozens of machines around the world. She was also the first with mobility support for Mobile Crowdsourcing systems. Additionally, this thesis contributes to the implementation of MktPoints, which is a new incentive mechanism linked to a reputation service. Inspired by network marketing and the psychological effects of variable rewards, MktPoints does not compromise system scalability and acts to attract/retain workers without necessarily involving extrinsic motivators.

Keywords: Crowdsourcing, Future Internet, Scalability.

Lista de Figuras

2.1 Exemplo do esgotamento da capacidade de expansão que cerca os esquemas de pirâmides 17

4.1 Arquitetura Snow Flurry 46

5.1 Implementação da arquitetura Snow Flurry I e indicação das etapas dos processos, da atribuição de tarefa ao recebimento dos resultados 54

5.2 Visualizando a área de atuação do trabalhador 54

5.3 Diagrama da criação de tarefa por ponto de interesse 56

5.4 Caixa de diálogo que ilustra a criação de uma tarefa com definição de contexto (neste exemplo, a cor verde) 57

5.5 Diagrama da criação de tarefa por área de interesse 58

5.6 Notificando sem usar a área de atuação dos trabalhadores 59

5.7 Visão do solicitante no aplicativo 60

5.8 Visão do trabalhador no aplicativo 61

5.9 Visualização da resposta, com foto, na página *web* 62

5.10 Diagrama ilustrando o processo de recebimento da tarefa e envio dos resultados por parte do trabalhador 63

5.11 Implementação da arquitetura Snow Flurry II e indicação das etapas dos processos, incluindo a comunicação direta entre dispositivos 64

7.1 Avaliação da latência dos componentes utilizados na implementação da arquitetura Snow Flurry I 81

7.2 Avaliação do tempo de resposta do Auspice na etapa de geolocalização dos trabalhadores 85

7.3 Deslocamento da média e aumento da variância no tempo de resposta do Auspice na etapa de geolocalização dos trabalhadores 85

7.4 Avaliação das duas principais etapas de comunicação do MktPoints, o pedido de reserva e o registro do MOS 87

7.5 Deslocamento da média e aumento da variância no tempo de resposta (MOS com Java Socket) 88

7.6 Deslocamento da média e aumento da variância no tempo de resposta (Reserva com Java Socket) 89

7.7 Deslocamento da média e aumento da variância no tempo de resposta (MOS com Node.js) 90

7.8 Deslocamento da média e aumento da variância no tempo de resposta (Reserva com Node.js) 91

7.9 Avaliação da capacidade de atendimento das implementações com Java e NodeJS do MktPoints 92

Lista de Tabelas

4.1 Comparativo entre arquiteturas 52

5.1 Linhas de código utilizadas na implementação das arquiteturas Snow Flurry I e II, com MktPoints 59

6.1 *Mean Opinion Score - MOS* 68

6.2 Exemplo de pontuação com MOS 69

6.3 Exemplo de distribuição dos pontos através dos níveis 70

6.4 Legendas da Tabela 6.5 74

6.5 Comparativo entre mecanismos de incentivo 76

6.6 Comparativo entre mecanismos de incentivo - continuação 77

6.7 Comparativo entre serviços de reputação 78

6.8 Legendas da Tabela 6.9 79

6.9 Comparativo entre sistemas de recomendação 79

7.1 Avaliação da latência e ganhos derivados dos componentes utilizados na implementação da arquitetura Snow Flurry I 82

Lista de Abreviaturas e Siglas

ACL	:	Access Control List;
ALG	:	Application-Layer Gateway;
AP	:	Access Point;
API	:	Application Programming Interface;
APNS	:	Apple Push Notification Service;
App	:	Abreviação da palavra inglesa application, no contexto desta tese significa programa desenvolvido para smartphones e/ou tablets;
CAPTCHA	:	Completely Automated Public Turing test to tell Computers and Humans Apart;
CDD	:	Comunicação Direta entre Dispositivos;
CCN	:	Content Centric Networking;
CORS	:	Cross Origin Resource Sharing;
CPU	:	Central Processing Unit;
D2D	:	Device-to-Device;
DDoS	:	Distributed Denial of Service;
DNS	:	Domain Name System;
DTN	:	Delay-Tolerant Networking;
FIA	:	Future Internet Architecture;
FTP	:	File Transfer Protocol;
GC	:	Garbage Collection;
GNS	:	Global Name Service;
GPS	:	Global Positioning System;
GSM	:	Global System for Mobile Communications;
GUID	:	Globally Unique Identifier;
HTML	:	HyperText Markup Language;
HTTP	:	Hypertext Transfer Protocol;
ICE	:	Interactive Connectivity Establishment;
ICMP	:	Internet Control Message Protocol;

ICN	:	Information Centric Networking;
IETF	:	Internet Engineering Task Force;
IP	:	Internet Protocol;
MCS	:	Mobile Crowdsourcing ou, dependendo do contexto, Mobile Crowd Sensing;
MCTA	:	Maximum Complex Task Assignment;
MIPv6	:	Mobile IPv6;
MOS	:	Mean Opinion Score;
MPNS	:	Microsoft Push Notification Service;
MSA	:	Maximum Score Assgnment;
MTA	:	Maximum Task Assgnment;
MTurk	:	Amazon Mechanical Turk;
NAT	:	Network Address Translation;
NETLMM	:	Network-based Localized Mobility Management;
NoSQL	:	Banco de dados orientado a documentos;
NSF	:	National Science Foundation;
OCR	:	Optical Character Recognition;
P2P	:	Peer-to-Peer;
PMIPv6	:	Proxy Mobile IPv6;
PRISM	:	Platform for Remote Sensing using Smartphones;
PoC	:	Proof of Concept;
PoI	:	Point of Interest;
QoI	:	Quality of Information;
RDB-SC	:	Reliable Diversity-Based Spatial Crowdsourcing;
RTP	:	Real-time Transport Protocol;
SANE	:	Server Access Network Entities;
SBC	:	Session Border Control;
SC	:	Spatial Crowdsourcing;
SDN	:	Software Defined Networking;
SHEs	:	Smart Home Environments;
SIP	:	Session Initiation Protocol;
SPS	:	Social Participatory Sensing;
SQL	:	Structured Query Language;
SR	:	Serviço de Reputação;

STUN	:	Session Traversal Utilities for NAT;
TCP	:	Transmission Control Protocol;
TFTP	:	Trivial File Transfer Protocol;
TURN	:	Traversal Using Relay NAT;
UDP	:	User Datagram Protocol;
UMass	:	University of Massachusetts;
UPnP	:	Universal Plug and Play;
VoIP	:	Voice over Internet Protocol;
Wi-Fi	:	Wireless Fidelity;
WSN	:	Wireless Sensor Networking;

Sumário

1 Introdução **1**

1.1 Objetivos 4

1.2 Contribuições 5

1.3 Demonstrações e resultados 5

1.4 Estrutura 5

2 Fundamentação teórica **6**

2.1 *Crowdsourcing* 6

2.1.1 Transmissão de dados brutos versus pré-processamento na origem . 7

2.1.2 Segurança 8

2.1.3 Mecanismos de incentivo 8

2.1.4 Limitação de recursos 9

2.1.5 Arquitetura 9

2.1.6 Escalabilidade 10

2.2 Mobilidade 11

2.2.1 IP Móvel 12

2.2.2 Redes definidas por *software* 12

2.2.3 Redes orientadas por conteúdo 13

2.2.4 Soluções baseadas em rede ou em dispositivos 13

2.2.5 Serviço global de nomes 14

2.3 Comunicação direta entre dispositivos 15

2.4 Marketing de rede 16

2.4.1 Plano de compensação 16
2.4.2 Comissão 18
2.4.3 Lucratividade 18
2.4.4 Aplicação em *crowdsourcing* 19

3 Trabalhos relacionados 20

3.1 Escalabilidade 20
3.1.1 Algoritmos 20
3.1.2 Computação em nuvem 24
3.1.3 *Proxy* 24
3.1.4 Redes tolerantes à atrasos 25
3.1.5 Comunicação direta em redes 5G 25
3.1.6 Virtualização e suporte à mobilidade 25
3.1.7 Computação de borda 26
3.1.8 Medusa e a computação elástica 26
3.2 Travessia de NAT 27
3.2.1 Protocolo STUN 27
3.2.2 Protocolo SIP 27
3.2.3 Outras abordagem e atravessando *firewalls* 29
3.2.4 Conexão por TCP 29
3.2.5 Utilizando SDN 31
3.2.6 Internet do futuro 31
3.3 Envolvimento e qualidade 32
3.3.1 Mecanismos de incentivo 32
3.3.1.1 Maximização da utilidade, verdade e a qualidade 32
3.3.1.2 Racionalidade individual e o bem-estar social 33
3.3.1.3 Eficiência computacional 34

3.3.1.4 Arrependimento 35
3.3.1.5 Redução de custos 36
3.3.1.6 Recompensas variáveis 38
3.3.1.7 Gamificação 39
3.3.1.8 Marketing de rede 40
3.3.1.9 Considerações 41
3.3.2 Sistemas de reputação 41
3.3.2.1 Confiança versus qualidade 41
3.3.2.2 Seleção por credibilidade calculada 42
3.3.2.3 Penalidades 43
3.3.2.4 Atenção aos prazos e redes sociais 43
3.3.2.5 Solução distribuída 44

4 Arquitetura Snow Flurry 45

4.1 Elementos da arquitetura 45
4.1.1 Serviço de notificação 45
4.1.2 Retaguarda 47
4.1.2.1 Servidores geograficamente distribuídos 47
4.1.2.2 Algoritmo de consenso 47
4.1.2.3 Algoritmo de pesquisa em banco de dados distribuído . . . 48
4.1.2.4 Servidores com NoSQL 48
4.1.2.5 Suporte à contexto 48
4.1.2.6 Algoritmo de geolocalização 49
4.1.2.7 Suporte à mobilidade 49
4.1.3 Dispositivo móvel 49
4.1.3.1 Recursos para atendimento das tarefas 49
4.1.3.2 *Interface* do solicitante no *app* 50

4.1.3.3 Banco de dados local com SQL 50
4.1.3.4 Módulo de segurança 50
4.1.4 Comunicação direta entre dispositivos 50
4.1.5 Rede de distribuição de conteúdo 51
4.1.6 Serviço de reputação 51
4.2 Comparativo entre arquiteturas do estado da arte 51

5 Implementação da Snow Flurry I 53

5.1 Componentes 53
5.2 O processo 53
5.3 Segurança 59
5.4 Prova de conceito 61
5.5 Snow Flurry II e o serviço Espelho UDP 62
5.5.1 MSocket 64
5.5.2 Qual o custo da comunicação direta para os usuários? 64
5.5.3 Alternativas: Mobile P2P e nuvem pública 65
5.6 Snow Flurry III e o Auspice 65

6 MktPoints: A implementação do mecanismo de incentivo e do serviço de reputação 67

6.1 Qualificação das respostas 67
6.2 Gamificação 68
6.3 Marketing de rede 69
6.4 Caixa de Skinner 70
6.5 Considerações sobre *truthfulness* 71
6.6 Ataque Sybil 71
6.7 Comparativo entre mecanismos de incentivo do estado da arte 73
6.8 Comparativo entre serviços de reputação do estado da arte 74

6.9 Comparativo entre sistemas de recomendação do estado da arte 75

7 Avaliação **80**

7.1 A prova de conceito e o experimento comparativo com o Medusa 80

7.1.1 Descrição do cenário 82

7.1.2 Análise dos resultados 82

7.1.2.1 Complexidade 83

7.1.2.2 Custo 83

7.1.3 Recomendações à UMass 84

7.2 Observações complementares em experimentos de escalabilidade com o MktPoints 85

7.2.1 Descrição do cenário 85

7.2.2 Análise dos resultados 86

7.3 Por que remover o atraso de rede? 92

8 Conclusão **93**

8.1 Limitações 94

8.2 Trabalhos futuros 94

Referências **96**

Capítulo 1

Introdução

A palavra *crowdsourcing* foi utilizada pela primeira vez por Jeff Howe e Mark Robnson, editores da Revista Wired. Eles fizeram uma composição com as palavras *crowd* (multidão) e *outsourcing* (terceirização) para nomear sistemas que orquestram a colaboração massiva de indivíduos para solução de algum tipo de problema ou realização de uma tarefa [89].

Uma história envolvendo o teste público completamente automatizado de Turing para diferenciar computadores e seres humanos (CAPTCHA, do termo em inglês *Completely Automated Public Turing test to tell Computers and Humans Apart*) ilustra um exemplo de uso dos sistemas de *crowdsourcing*. O CAPTCHA foi desenvolvido para evitar sobrecargas causadas por acessos automatizados. Para atingir esse objetivo, ele testa o usuário, verificando se quem está acessando o site é um humano ou um *software*. Essa verificação é feita por meio de uma tarefa que só humanos conseguem executar com sucesso. A tarefa envolve a digitação de uma palavra que aparece na tela, cujo formato foge à capacidade de interpretação dos programas de reconhecimento óptico de caracteres (OCR, do termo em inglês *Optical Character Recognition*). Aproveitando esse cenário, Luis von Ahn resolveu criar o projeto reCAPTCHA que tem o objetivo de ajudar na digitalização de livros antigos. Quando o OCR não reconhece alguma palavra, o sistema a envia ao reCAPTCHA para que seu reconhecimento seja feito por humanos durante um processo similar ao CAPTCHA. No novo método, duas palavras são apresentadas, uma que permite a entrada no sistema e a outra que não passou pelo OCR. O usuário é autorizado à seguir em frente se acertar a palavra do CAPTCHA, já a resposta relativa a outra palavra é apenas armazenada e comparada com as respostas de outros usuários. Existindo um consenso, considera-se a palavra interpretada pelos usuários. Dessa forma, milhares de livros foram

digitalizados num curto espaço de tempo em uma operação impossível de ser realizada por máquinas e demorada demais para ser feita por uma equipe de humanos [189].

Outro exemplo do poder das multidões é o site Yahoo Respostas[1], nele o usuário pode fazer perguntas e se propor a responde-las em troca de pontos. Os pontos ganhos são utilizados para fazer perguntas dentro do sistema, mas existem sistemas de *crowdsourcing* que trabalham com pagamento monetário e o *Amazon Mechanical Turk* (MTurk)[2] é um deles. Esse sistema permite a requisição de vários tipos de tarefas aos usuários. Escolher a melhor entre várias fotografias de uma loja, redigir descrições de produtos ou identificar artistas em CDs de música são alguns exemplos.

Mobile crowdsourcing (MCS) e *spatial crowdsourcing* (SC) são termos que surgiram depois da popularização dos dispositivos móveis e se referem a sistemas que possibilitam uma atribuição de tarefas com restrições de tempo e espaço aos usuários desses equipamentos. Um exemplo de requisição é a localização de vagas para estacionar, em que as pessoas podem facilmente identificar locais disponíveis e fazer um relatório com imagens e/ou mensagem de texto. Se fôssemos utilizar um sistema autônomo para localizar vagas, baseado em ultrassom, não só precisaríamos de *hardware* especial, mas também de algoritmos sofisticados para garantir a confiabilidade dos dados [77]. Como alguns termos mudam de um trabalho para o outro, vamos chamar de trabalhadores os usuários que respondem as tarefas dos solicitantes. Logo, os solicitantes são os usuários que criam as tarefas no MCS. Chamaremos de campanha a criação de uma tarefa (ou conjunto de tarefas relacionadas) e sua distribuição a mais de um trabalhador.

Se comparado com a tradicional rede de sensores sem fio (WSN, do termo em inglês *Wireless Sensor Networking*), o MCS possui muitas vantagens, como mobilidade[3], escalabilidade, custo-benefício e inteligência humana [74]. Entre alguns casos de uso existentes, podemos citar:

1. Atendimento de emergência [4];
2. Definição de rota em situações de crise [5] e serviço logístico [10];
3. Pesquisa científica [13];
4. Jornalismo [6, 9];

[1] https://br.answers.yahoo.com/
[2] https://www.mturk.com/
[3] Neste contexto, mobilidade representa a possibilidade de acionar sensores de dispositivos móveis como, por exemplo, *smartphones* e *tablets*.

5. Controle [18] e compartilhamento de preços [22];
6. Pesquisa de mercado [19] e questionários georeferenciados [132];
7. Resumo de opiniões sobre aplicativos [17];
8. Mapeamento do interior de edifícios [42];
9. Cidades inteligentes [25, 26];
10. Mapeamento de criminalidade [21];
11. Fiscalização de transito [23];
12. Apoio aos cidadãos com mobilidade[4] reduzida [33];
13. Localização de crianças perdidas [1];
14. Assistência direta a pessoas em situação de extrema pobreza [3] e serviços humanitários [16];
15. Estacionamento inteligente e combate a *free-riders* [45].

Sistemas MCS enfrentam alguns desafios, entre os quais destacamos a escalabilidade. Definimos esse desafio como a capacidade de atender a uma demanda crescente sem perda de performance, ponto ignorado em muitas propostas que fazem uso do modelo tradicional (cliente-servidor). Nesse modelo, a retaguarda (*backend*) é responsável pelo cadastro dos usuários, distribuição das tarefas, recepção das respostas e, em alguns casos, consolidação das informações. Porém, esse tipo de sistema não escala, ele perde performance, podendo parar, quando um grande número de usuários e tarefas chegam num curto espaço de tempo. No estado da arte encontramos soluções envolvendo balanceamento de carga com replicação ativa de servidores, *proxy*, DTN, virtualização, D2D e computação em nuvem, elástica ou de borda. Entretanto, elas apresentam limitações de ordem técnica e financeira.

A escalabilidade é um desafio para diversos tipos de sistemas e entre as pesquisas sobre esse tema destacamos os trabalhos da Universidade de Massachusetts (UMass) no desenvolvimento do serviço global de nomes (GNS, do termo em inglês *Global Name Service*) da arquitetura MobilityFirst, que é parte do programa da *National Science Foundation* para Internet do Futuro. O GNS viabiliza (1) um identificador único global (GUID,

[4]Neste contexto, mobilidade é a capacidade do indivíduo realizar movimentos.

do termo em inglês *Globally Unique Identier*) para cada usuário, (2) mobilidade[5] e (3) busca por contexto (como, por exemplo, localizar apenas taxistas) [167, 186, 148]. Ele foi desenvolvido com foco num cenário futuro onde estima-se a existência de bilhões de identidades móveis mudando o endereço de rede, pelo menos, dezenas de vezes por dia. Este GNS (também chamado de Auspice) é escalável e resolve identidades para localizações de redes, mesmo que ocorra mobilidade [186]. A arquitetura de rede MobilityFirst se destina a abordar diretamente os desafios de acesso sem fio e mobilidade em escala, além de fornecer novos serviços necessários a cenários emergentes voltados para aplicações de Internet móvel [130, 148]. Considerando que os sistemas de MCS estão voltados para esses cenários emergentes, podemos formular a seguinte hipótese:

> *Se um sistema MCS for composto por serviços escaláveis em todas as suas partes, então ele será escalável.*

1.1 Objetivos

O objetivo principal desta tese é uma arquitetura escalável, eficiente, simples, econômica e que oferece suporte à mobilidade. Para testar a viabilidade da arquitetura proposta, intitulada Snow Flurry[6], implementamos um MCS e realizamos uma prova de conceito (PoC, do termo em inglês *Proof of Concept*) que passa por quatro pontos principais: (1) cadastro dos usuários; (2) seleção dos usuários que irão realizar as tarefas; (3) atribuição de tarefas; e (4) recebimento das respostas. Para resolver os dois primeiros, utilizamos o Auspice. No terceiro item descentralizamos a carga com uma solução distribuída que trata o evento no programa local (*frontend*) do solicitante. Em seguida, no quarto item optamos pelos serviços do Google Firebase[7]: (1) o *Realtime Database* para armazenamento das respostas num banco de dados e (2) o *Cloud Storage* para recepção de arquivos. Também utilizamos o *Cloud Messaging* do Firebase para notificar os usuários.

Entre os objetivos específicos estão: (1) discutir a comunicação direta entre dispositivos e o problema da "Travessia de NAT" e (2) apresentar o MktPoints que é um novo mecanismo de incentivo para sistemas MCS.

Na lista dos objetivos técnicos estão: (1) o desenvolvimento de uma aplicação (App, abreviação da palavra inglesa *application*) para Android que possa receber tarefas, en-

[5]Neste caso e ao longo desta tese, mobilidade é a capacidade do dispositivo trocar de identificador sem perder as conexões estabelecidas.

[6]Inspirado nos móbiles de mesmo nome do escultor Alexander Calder.

[7]https://firebase.google.com/

caminhar as respostas e fazer solicitações; (2) o desenvolvimento de um sistema *web* de controle que possa fazer solicitações e visualizar as respostas; e (3) a implementação do Serviço de Reputação (SR) que é a base do novo mecanismo de incentivo.

1.2 Contribuições

Esse trabalho contribui com (1) uma nova arquitetura para MCS; (2) uma implementação da arquitetura que utiliza o Auspice para garantir escalabilidade e mobilidade ao sistema; (3) uma discussão sobre a comunicação direta entre dispositivos como uma alternativa ao uso das nuvens; e (4) um novo mecanismo de incentivo e a respectiva avaliação do mesmo em termos de escalabilidade.

1.3 Demonstrações e resultados

Além disso, a realização da *PoC* mostrou a viabilidade da arquitetura Snow Flurry. Já as análises de desempenho, volume, custo e complexidade mostraram que: a arquitetura oferece uma redução entre 26,31% e 32,03% na latência quando comparada ao estado da arte; ela dispensa o monitoramento e manutenção de uma infraestrutura globalmente distribuída de aproximadamente 200 máquinas; promove uma economia anual que pode chegar à US$ 10.000,00; e o risco de uma alta demanda comprometer o SR é de apenas 0,007%. Todas essas fatores ocorrem junto com o atendimento dos requisitos para uma arquitetura escalável. Por fim, a Snow Flurry também é a primeira arquitetura para MCS, no limite do nosso conhecimento, a oferecer suporte à mobilidade.

1.4 Estrutura

Revisamos os fundamentos das principais tecnologias envolvidas no Capítulo 2. Na sequência, apresentamos o estado da arte por meio dos trabalhos relacionados no Capítulo 3. O Capítulo 4 detalha de forma conceitual a arquitetura. Em seguida, o Capítulo 5 apresenta uma implementação e discute as escolhas feitas durante o projeto. Depois, no Capítulo 6 apresentamos um novo mecanismo de incentivo para, em seguida, expor no Capítulo 7 os cenários utilizados nos experimentos e os resultados obtidos. Por fim, o Capítulo 8 resume as conclusões e os trabalhos futuros.

Capítulo 2

Fundamentação teórica

No decorrer deste capítulo veremos mais informações sobre os pontos que formam a base deste trabalho, iniciando com as oportunidades e desafios enfrentados nas pesquisas sobre *crowdsourcing* e *crowd sensing* (devido as similaridades entre os dois) [35, 139]. Por último, apresentaremos um pouco mais sobre as pesquisas com o Auspice.

2.1 *Crowdsourcing*

Um ano antes de Jeff Howe publicar um artigo com o termo *crowdsourcing* [89], Luis von Ahn utilizou o termo "computação humana" em sua tese de doutorado [188]. Nesse trabalho podemos ver que, apesar dos avanços da inteligência artificial, apenas a mente humana é capaz de realizar as operações de síntese, contexto e reflexão. Logo, existem muitas tarefas que são facilmente executadas por pessoas e desafiadoras para os computadores mais sofisticados que existem. Na busca por um paradigma que resolvesse essas tarefas, surge o que hoje é mais conhecido como *crowdsourcing*.

Além dos trabalhos que tratam de *crowdsourcing*, vamos citar pesquisas envolvendo *crowd sensing* devido aos pontos em comum encontrados nas arquiteturas desses sistemas. Quanto as diferenças entre esses paradigmas, Ra et al. [145] destacam que o primeiro permite a solicitação de tarefas que necessitam da inteligência humana para a sua resolução, enquanto o segundo apenas capta dados de sensores (câmera, acelerômetro, microfone, GPS, etc). Para ilustrar como a diferença entre os conceitos pode ser sutil, observamos que, nesse mesmo trabalho, os autores apresentam um exemplo de *crowd sensing* que envolve a captação de vídeo, porém a tarefa requer que o usuário ligue a câmera e faça o vídeo. A solicitação de intervenção do usuário é feita por meio de uma notificação. Como, neste caso, não foi solicitada ao usuário a realização de uma tarefa intelectualmente com-

plexa, a atividade é definida como *crowd sensing*. Em alguns casos encontramos o termo *participatory sensing*, quando existe a necessidade de participação do usuário para a realização da tarefa. O *survey* de Christin et al. [50] é um exemplo, nele os autores relatam o funcionamento do *LiveCompare*. Esse sistema compara o preço de diversos produtos. Para alcançar esse objetivo, ele requer que o usuário tire fotos dos códigos de barras para coleta dos preços. Nesta tese, consideramos *crowd sensing* e *participatory sensing* sinônimos. Independente dos termos envolvidos, esses paradigmas enfrentam alguns desafios em comum. Vamos dedicar o restante deste capítulo à apresentação dos mesmos.

2.1.1 Transmissão de dados brutos versus pré-processamento na origem

Em alguns casos é interessante para ambas as partes (solicitantes e trabalhadores) um pré-processamento dos dados obtidos. Por exemplo, considere uma coleta regular de cinco segundos de áudio. Agora, imagine que no final da coleta temos uma hora de áudio (dados brutos). Suponha que o interesse do solicitante seja saber o maior nível registrado em decibéis em cada coleta de cinco segundos [77, 35].

Diante do exposto temos duas opções: (1) transmitir os dados brutos ao solicitante ou (2) realizar um pré-processamento desses dados e só enviar ao solicitante o maior nível em decibéis de cada coleta. Do ponto de vista do trabalhador, o custo do processamento pode ser menor se comparado ao custo da transmissão. Para o solicitante, reduz-se a quantidade de processamento e evita-se uma sobrecarga na retaguarda provocada por diversos trabalhadores enviando dados brutos ao mesmo tempo e consumindo toda a banda disponível do solicitante [77, 35].

Além do "conflito de escolha" entre processar e transmitir, existe o problema da "explosão" de análises. Como a prática corrente é desenvolver pré-processamentos exclusivos para cada aplicativo, existe o risco do atendimento simultâneo de tarefas quando muitas aplicações coexistem. Esse evento leva o dispositivo do trabalhador à exaustão dos recursos. Note que esses *apps* podem até usar o mesmo sensor e realizar cálculos similares [77, 35].

As necessidades de processamento sobre um mesmo tipo de dado pode variar muito de solicitante para solicitante, mas em geral o desafio de identificar padrões em grande quantidade de dados envolve determinados algoritmos de análise. Dependendo da quantidade de entrada e da tolerância ao atraso das aplicações ao processamento final dos dados, há duas abordagens possíveis: tradicional que armazena os dados e o modelo que trata o

fluxo de dados. Na tradicional, esses são armazenados numa base e depois aplicam-se vários algoritmos de mineração sobre o banco de dados para detectar padrões. No entanto, se a entrada contínua é excessiva para armazenamento ou o aplicativo exige uma rápida detecção de padrões, algoritmos de fluxo de dados (do termo em inglês, *data streaming algorithms*) podem ser necessários. Tais algoritmos usam o fluxo de dados para identificar padrões, sem a necessidade de primeiro armazená-los [77, 35].

2.1.2 Segurança

Ao instalar um *app* que requeira acesso aos diversos sensores de um *smartphone*, o usuário pode estar fragilizando a segurança do seu dispositivo. Sem uma proteção adequada, podem viabilizar o acesso a informações privadas dos seus proprietários. Os invasores podem tirar fotos da vida particular do usuário, traçar o caminho que ele faz habitualmente, monitorar os lugares visitados, etc. Muitas pessoas relutam em contribuir com sistemas de *crowdsourcing/crowd sensing*, com receio dessas consequências [103, 104, 50, 74].

Outro problema é o fornecimento de dados sensoriais errôneos (por exemplo, leituras de GPS forjadas) por indivíduos mal-intencionados. Manter a integridade dos dados sensoriais obtidos é fundamental para o sucesso dessas aplicações [77, 50, 74].

2.1.3 Mecanismos de incentivo

Aqui o desafio é elaborar mecanismos que atraiam trabalhadores que possam executar com qualidade as tarefas terceirizadas pelos solicitantes. Existem diversos tipos de incentivos. Alguns sistemas utilizam retribuições financeiras, mas isso não garante qualidade às soluções. Outros apelam para a satisfação pessoal do trabalhador, mas isso nem sempre atrai um número significativo de participantes [197, 193, 93].

Tanto as aplicações voltadas para *crowdsourcing* quanto as de *crowd sensing* enfrentam o problema da baixa adesão de trabalhadores. Isso ocorre por conta da sensação de insegurança e do consumo de recursos, tais como bateria e poder de processamento. Muitos pesquisadores têm estudado meios de motivar os usuários a contribuir com seus recursos, nesse campo encontramos os mecanismos de incentivo. Basicamente, eles fornecem certas recompensas à participação por meio de um modelo conhecido como leilão reverso. Normalmente, no leilão tradicional podemos oferecer um produto e estipular um preço inicial. Em seguida, solicitamos a um grupo de interessados que façam suas ofertas e leva o produto quem oferecer o maior valor. No leilão reverso podemos expor uma

necessidade que pode ser um produto ou serviço. Na sequência, solicitamos a um grupo de interessados que façam suas ofertas e o fornecedor será aquele que oferecer o menor valor [197, 193, 93].

Devido às restrições de tempo e espaço que cercam as tarefas em MCS, consideramos que o leilão reverso não é o modelo mais adequado para esse tipo de sistema. Diante dessa inferência, buscamos alternativas em outras áreas do conhecimento para fomentar uma nova proposta e chegamos ao marketing de rede. Mais a frente, veremos o que é marketing de rede antes de discutir o uso desse modelo de negócio no desenvolvimento de um novo mecanismo de incentivo.

2.1.4 Limitação de recursos

Quando dispositivos móveis participam do sistema, devemos ter em mente que o uso primário deles deve ficar reservado ao uso cotidiano. Os trabalhadores só concordarão em contribuir usando, por exemplo, seus *smartphones*, se o processo não fizer uso significativo dos recursos [103, 104]. Isso cria um "conflito de escolha" entre qualidade e limitação de recursos. Existem diferentes tipos de dados que podem ser utilizados para um mesmo efeito, mas com diferentes níveis de qualidade e consumo de recursos. Aproveitar essas diferenças para melhorar a qualidade, minimizando o consumo de recursos é um novo desafio. Por exemplo, os dados de localização podem ser fornecidos usando GPS, Wi-Fi e GSM, com diferentes níveis de precisão. Em comparação com Wi-Fi e GSM, a contínua amostragem da localização com GPS é a que fornece maior precisão espacial, mas também é a que consome a maior quantidade de energia [77, 35].

Outra questão está na existência de múltiplos aplicativos, ativos simultaneamente, no dispositivo do trabalhador. Isso complica a alocação de recursos, pois cria uma competição por sensores entre diferentes aplicativos. Mesmo quando os *apps* trabalham com priorização de tarefas ainda há competição, a prioridade que um pedido exige de determinado sensor pode reduzir a taxa de amostragem de outros sensores ou torná-los indisponíveis [77, 35].

2.1.5 Arquitetura

Geralmente, uma arquitetura para *crowdsourcing* ou *crowd sensing*, possui uma retaguarda e uma aplicação no dispositivo do trabalhador. O solicitante costuma utilizar uma aplicação hospedada na retaguarda, que é responsável pela distribuição das tarefas

e sumarização dos resultados coletados [77, 35]. O fluxo desse modelo padrão está listado a seguir:

1. Primeiro o solicitante cria uma tarefa utilizando um programa local, normalmente um *website*, que a encaminha à retaguarda da plataforma;
2. Em seguida, a retaguarda atribui a tarefa aos trabalhadores que atendem a determinados requisitos;
3. De tempos em tempos, os trabalhadores verificam, normalmente por meio de um *app*, a existência de tarefas junto à retaguarda;
4. Ao concluir as tarefas, os trabalhadores enviam os dados à retaguarda;
5. Após o prazo limite estabelecido na tarefa, à retaguarda consolida as informações;
6. Por último, o solicitante acessa o programa local que obtém as informações consolidadas junto à retaguarda.

Em alguns casos, a retaguarda protege a identidade dos solicitantes e trabalhadores por meio de passos adicionais que envolvem o uso de uma camada intermediária de segurança. Também existem casos em que o processo termina com o solicitante atribuindo uma nota de qualidade às entregas dos trabalhadores [77, 35].

Esse modelo impede o desenvolvimento e implantação de novos aplicativos de várias maneiras. Para escrever um novo, o desenvolvedor tem que enfrentar os desafios em matéria de energia, privacidade e qualidade dos dados de forma *ad hoc*. Além disso, ele pode necessitar de diferentes variantes de pré-processamento (vide Seção 2.1.1), se ele quiser executar o aplicativo em dispositivos heterogêneos (sensores que produzem dados com níveis de qualidade diferentes) utilizando diferentes sistemas operacionais [77, 35].

2.1.6 Escalabilidade

Um dos principais desafios para as arquiteturas atuais é a escalabilidade. A arquitetura tradicional, apresentada na Seção 2.1.5, não escala. Entre as causas podemos citar: (1) a sobrecarga causada na retaguarda no momento do recebimento das respostas; (2) o consumo de recursos na retaguarda derivado da consolidação das informações; (3) a manutenção de milhões de contas pertencentes aos solicitantes e do tratamento do trafego

resultante; e (4) a coexistência de diversas aplicações no dispositivo do trabalhador, concorrendo por recursos. Nesse último caso, segundo Ganti, Ye e Lei [77], existem propostas que buscam uma plataforma genérica que centralizam todos os tipos de tarefas em um único *app*, mas eles acreditam que isso tira a flexibilidade. Para contornar o problema da falta de flexibilidade, alguns autores propuseram um esquema que instala algoritmos adicionais na aplicação do trabalhador. Procedimento semelhante ao uso de *plugins* nos navegadores [145].

Mesmo com esse novo esquema, a escalabilidade ainda é um problema na retaguarda e na camada de segurança. A retaguarda precisa tratar a proximidade dos trabalhadores, o gerenciamento das contas, a sobrecarga no recebimento das repostas, o desempenho nas pesquisas sobre a base de dados e na consolidação dos dados. Quanto a camada de segurança, ela promove uma computação intensiva ao utilizar técnicas criptográficas para transformar os dados, a fim de, preservar a privacidade. Além disso, ela não é escalável porque essas técnicas exigem a geração e manutenção de várias chaves, o que também leva a um maior consumo de energia. Alguns pesquisadores têm utilizado técnicas de perturbação de dados, no lugar da criptografia [77, 35].

2.2 Mobilidade

O serviço de endereçamento da camada de rede é definido pelo IP. Nesse esquema, cada endereço IP contém grupos onde parte identifica a rede e a outra identifica o dispositivo. Os roteadores existentes em todo o mundo têm tabelas de roteamento que informam qual caminho deve ser usado para se chegar à determinada rede. Quando um dispositivo móvel deixa a conexão com um ponto de acesso (AP, do termo em inglês *Access Point*) de uma rede sem fio e se conecta a outro AP, de outra rede, ele ganha um novo endereço IP. Logo, os pacotes transmitidos para o antigo IP, e direcionados à antiga rede, não serão recebidos pelo dispositivo em seu novo "endereço". Para manter a conexão, muitos programas precisariam ser informados da mudança. Uma solução seria fazer com que os roteadores utilizassem tabelas de dispositivos (usando o IP inteiro como identificador) ao invés de tabelas de redes, mas isso teria um alto custo em virtude do tamanho necessário para seu armazenamento e ao tempo necessário para processamento [179].

2.2.1 IP Móvel

Outra solução seria partir do princípio que todas as máquinas têm um local inicial permanente, que nunca muda. Dividir o mundo geograficamente em pequenas áreas e nomear um ou mais "agentes" para cada uma. Um agente pode ser do tipo "externo", quando controla a visita de dispositivos móveis, e/ou "local", quando controla o registro dos dispositivos que declaram pertencer a área de atuação do mesmo. Quando um dispositivo chega a uma área ele deve se identificar. O agente externo entra em contato com o agente local do visitante para confirmar se ele é quem diz ser. Confirmado os dados, a identificação é aceita. Por fim, o agente externo cria uma entrada em suas tabelas e confirma o registro ao visitante [138, 179, 97]. Em resumo, essa é a solução visada pelo IP Móvel. Além do que foi citado, o IP Móvel usa um esquema de túneis para redirecionar os pacotes ao novo endereço [179]. Essa abordagem resolve diversos problemas, mas não é difícil notar que todo esse trabalho causa uma maior latência no restabelecimento do acesso e sobrecarga de informações de controle.

2.2.2 Redes definidas por *software*

Existem propostas que apostaram na separação da localização física da identificação, mas essas enfrentam problemas como escalabilidade e aplicabilidade [34]. Propostas recentes fazem uso de novas áreas de pesquisa (como as redes definidas por *software*) na tentativa de solucionar o problema da mobilidade [174, 27, 192]. O Projeto 4WARD [174, 27] mostra que o IP Móvel não seria um bom método devido à natureza dos dispositivos móveis atuais, cuja principal característica é a rápida movimentação entre redes. Tal movimentação acelerada poderia levar a altos atrasos ao criar túneis constantemente. Eles propõem uma arquitetura de Internet do Futuro que pretende virtualizar redes para que novos protocolos não precisem passar por padronizações demoradas, já que a virtualização é capaz de alterar as conexões e caminhos que passam pelas redes virtuais, realizando o mesmo trabalho dos agentes móveis. Além disso, a virtualização permitiria a implantação de novos serviços pelos provedores, sem depender de um padrão internacional, bastando apenas que os provedores trabalhem na sua rede virtual designada. Por fim, a proposta ainda seria rentável devido à possibilidade de se dividir recursos da infraestrutura de rede. Já no trabalho de Wang, Bi e Zhang [192], que também utilizou redes definida por *software* (SDN, do termo em inglês *Software Defined Networking*), os autores obtiveram bons resultados nos itens viabilidade, escalabilidade e alta capacidade de adaptação, mas as SDN ainda não são uma opção quando a questão é aplicabilidade. Xiao et al. [196]

chegou a idealizar uma solução com SDN, mas reconheceu que a proposta oferecia risco de sobrecarga de virtualização e não contemplava o suporte à mobilidade.

2.2.3 Redes orientadas por conteúdo

A aplicabilidade também afeta o trabalho de Luo et al. [118], onde se discute o uso das redes centrada em conteúdo (CCN, do termo em inglês *Content Centric Networking*), que são proeminentes redes centrada na Informação (ICN, do termo em inglês *Information Centric Networking*). A CCN, como outras arquiteturas ICN, baseia-se na ideia de nomear conteúdo em vez de dispositivos, permitindo o roteamento para caches de conteúdo popular [61, 62]. Tem sido demonstrado que a CCN pode suportar conversas ponto-a-ponto em tempo real como, por exemplo, as chamadas de voz ou vídeo. No entanto, esses experimentos não levaram em consideração a mobilidade dos dispositivos em tempo real. Nesse tema não há uma proposta voltada diretamente para *crowdsourcing*, mas podemos inferir, através do trabalho de Wang et al. [190] que tem o objetivo de melhorar o desempenho da CCN utilizando *crowdsourcing*, que uma solução com CCN precisaria vencer problemas com roteamento, cache de conteúdo e segurança.

2.2.4 Soluções baseadas em rede ou em dispositivos

Em Willkie, Lioy e Armstrong [195], vemos uma solução para mobilidade com base na rede, contrariando as abordagem baseadas em dispositivo. No trabalho de Kong et al. [109] o tema volta, pois um protocolo de gestão da mobilidade baseada em rede chamado Proxy Mobile IPv6 (PMIPv6) estava em processo de padronização pelo grupo de trabalho IETF NETLMM. Eles optaram por um caminho oposto aos protocolos existentes, tais como Mobile IPv6 (MIPv6), que são abordagens baseadas em dispositivo, mesmo quando, anos antes, Banerjee, Wu e Das [24] já haviam afirmado que embora os protocolos da camada de aplicação sejam piores do que os protocolos que operam nas camadas inferiores, em termos de atraso, transição e sobrecarga de sinalização, eles são mais adequados como uma solução de mobilidade potencial para a próxima geração de redes heterogêneas, se considerarmos fatores como a modificação da pilha de protocolos, a mudança de infraestrutura e a complexidade operacional inerente.

2.2.5 Serviço global de nomes

Em meio a tudo isso, podemos encontrar uma proposta que desvinculou os nomes dos IPs e que promete ser tão rápida quanto um sistema de nomes de domínio (DNS, do termo em inglês *Domain Name System*). DNS é o serviço da camada de aplicação que oferece uma rápida resolução de nomes. Ele atrela, de forma hierárquica, nomes a IPs [11, 179]. A nova proposta localiza dispositivos com um algoritmo desenvolvido para pesquisa em redes não hierárquicas, o Chord [179]. Basicamente, esse algoritmo consegue descobrir quem tem algum registro, se for o caso, sem um banco de dados central. O Chord é uma das peças fundamentais do Auspice, o GNS proposto na arquitetura MobilityFirst.

A arquitetura de rede MobilityFirst se destina a abordar diretamente os desafios de acesso sem fio e mobilidade em escala, além de fornecer novos serviços necessários para cenários emergentes voltados às aplicações de Internet móvel [130, 148]. Nesse contexto, o Auspice trabalha por meio do GUID. Com ele, pode-se salvar e recuperar informações. Para obter um GUID, basta criar uma conta no site do serviço[1] com um endereço de *e-mail* válido [166]. Para trabalhar com o MobilityFirst, também é necessário instalar e configurar um cliente Auspice, como também as chaves criptográficas e alterar o *software* do usuário, para que o mesmo utilize o MSocket (vamos detalhar esse recurso mais a frente). Outra opção ofertada pela arquitetura MobilityFirst é utilizar uma interface de programação para aplicativos (API, do termo em inglês *Application Programming Interface*) baseada em chamadas HTTP, para se comunicar com o Auspice. Esse foi o método que escolhemos para realização da PoC.

O Auspice foi desenvolvido com foco num cenário futuro onde estima-se a existência de bilhões de identidades móveis mudando o endereço de rede, pelo menos, dezenas de vezes por dia. Ele é altamente escalável e resolve rapidamente identidades para localizações de redes sobre alta mobilidade [186]. Conforme explicam Tie, Sharma e Venkataramani [181], o núcleo do Auspice é um motor de alocação que alcança os objetivos de latência, custo e disponibilidade, adaptando o número e locais das réplicas de cada registro de nome, conforme: (1) taxas de solicitações de pesquisa e de atualização para o nome; (2) geo-distribuição de pedidos para o nome; e (3) carga de requisições agregadas em todos os nomes.

[1] https://gns.name

2.3 Comunicação direta entre dispositivos

Nesta tese, vamos apresentar uma proposta de comunicação direta entre dispositivos que tem o objetivo de evitar os aportes financeiros e controles/configurações adicionais necessários à computação em nuvem e à computação elástica. Nela, o usuário que realizou a tarefa transmite os dados diretamente ao que criou a tarefa. Esse processo não é trivial, podemos encontrar na literatura algumas propostas sobre como resolver o que ficou conhecido por "problema da travessia de NAT".

A comunicação direta entre usuários finais, na Internet, enfrenta problemas devido à mobilidade e à escassez dos números de IP. Quando um equipamento troca de rede, o IP é alterado, afetando todo processo de endereçamento das aplicações correntes. Obter um IPv4 fixo é uma alternativa, porém cara e inviável se considerarmos os números disponíveis frente à quantidade de dispositivos móveis. O IPv6 resolve essa questão, mas não se encontra em pleno uso. Para contornar esses problemas, as redes atribuem números locais e utilizam a tradução do endereço de rede (NAT, do termo em inglês *Network Address Translation*) para viabilizar a comunicação dessas máquinas com a Internet. No entanto, o endereço IP recebido por elas é invisível às máquinas que estão fora da rede privada, ou seja, na Internet. No envio dos dados, o NAT troca o IP local pelo IP do servidor que tem conexão com a Internet e faz o processo inverso quando um dado chega. Por isso, serviços como o WhatsApp[2] dependem de um servidor para intermediar a troca de mensagens entre equipamentos, isto é, os dispositivos não se comunicam diretamente. Esse problema é conhecido como "Travessia de NAT" e existem algumas proposta para contorná-lo, normalmente utilizando o protocolo UDP [113, 155, 114, 191].

O protocolo UDP não deveria ser utilizado para fluxos de bytes confiáveis, tais como a transferência de arquivos. Porém, existem exceções à regra como o protocolo TFTP que foi feito para redes locais de alta confiabilidade. Na Internet, seu desempenho é muito inferior a sua versão confiável, o protocolo FTP. Outra opção é o protocolo utilitários transversais de sessão para NAT (STUN, do termo em inglês *Session Traversal Utilities for NAT*) que trabalha com muitos NATs sem requerer nenhum comportamento especial deles. A tecnologia 5G promete resolver essa questão no futuro com a tecnologia D2D [113, 155, 114, 191].

[2]https://www.whatsapp.com/?l=pt_br

2.4 Marketing de rede

Basicamente, o marketing de rede é um modelo de negócio onde seus representantes não recebem comissão apenas pelo que vendem. Eles também recebem pelas vendas das pessoas que recrutaram, direta e indiretamente. Logo, torna-se muito interessante convidar o maior número possível de pessoas e estimulá-las a fazer o mesmo sucessivamente.

O marketing de rede (também conhecido como marketing multinível) pode ser utilizado pelos sistemas de *crowdsourcing* como um mecanismo de incentivo à participação se realizarmos algumas modificações no processo, pois sua forma padrão não é lícita em dezenas de países. No Brasil, essa atividade é considerada crime contra a economia popular (Lei n° 1.521 de 26/12/1951) e equiparada aos antigos esquemas de pirâmides. O marketing de rede camufla a pirâmide vendendo produtos e serviços por meio do marketing direto. Esses métodos já causaram prejuízos, ao longo da história, a milhares de pessoas. Em 1998, ocorreu um dos piores eventos. Naquele ano, a Albânia experimentou um crescimento sem precedentes e um espetacular colapso desses esquemas. A população acusou o governo de conivência e, nos dias que se seguiram, o pais entrou em guerra civil. Mais de 2.000 pessoas morreram [95].

Esses sistemas quebram quando as últimas pessoas recrutadas não conseguem recrutar novos candidatos. Logo, elas deixam o esquema iniciando um efeito dominó que desfaz toda a pirâmide. Por exemplo, considere uma empresa de marketing de rede que comece com 6 pessoas compondo o primeiro nível. Agora imagine que cada uma convide outras 6 pessoas e que cada novo integrante faça o mesmo. Podemos ver na Figura 2.1 que para compor o 11° nível precisaríamos de mais pessoas que a população dos EUA. Se evoluirmos mais dois, a população do planeta não seria suficiente.

Na parte central de qualquer modelo sobre marketing de rede está o plano de compensação. Ele é quem controla toda a distribuição por meio de algoritmos e as implementações variam muito de empresa para empresa.

2.4.1 Plano de compensação

Tanto a área comercial quanto a acadêmica contam com milhares de publicações sobre marketing de rede [55]. Entretanto, quando o assunto é o sistema de compensação, tornam-se mais difícil encontrar alguma literatura que explique em detalhes as fórmulas ou algoritmos envolvidos. As empresas do ramo não têm o hábito de divulgar essa informação.

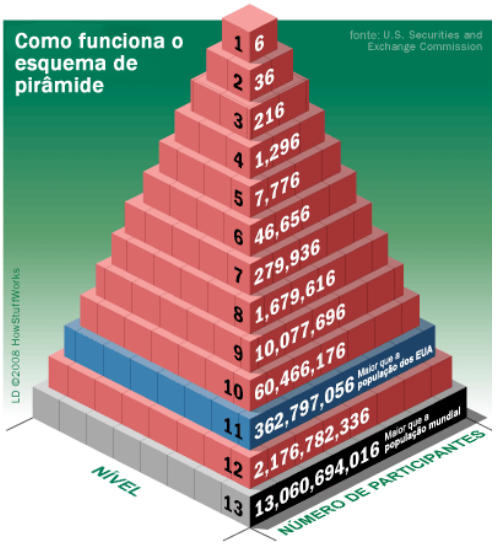


Figura 2.1: Exemplo do esgotamento da capacidade de expansão que cerca os esquemas de pirâmides

O plano de compensação, também conhecido como plano de remuneração ou plano de marketing, é uma das partes mais importantes do marketing de rede. Segundo Costa [53], existem milhares de empresas nessa atividade e cada uma delas tem um plano diferente designados como: Matriz (*Matrix*), Uninível (*Unilevel*), Binário (*Binary*), Australiano (*Australian Two-Up*), Emancipação Gradativa (*Breakaway/Stairstep*) e outras variações.

1. O plano Matriz tem lateralidade (número de convidados) e profundidade (níveis) limitadas. O tamanho potencial da rede é pré-definido pela companhia [53, 32].

2. O Uninível tem lateralidade infinita e profundidade limitada. Esse é o plano mais tradicional e até hoje permanece como um dos mais utilizados por ser de fácil compreensão [53, 32].

3. O Binário estimula a formação de apenas duas pernas em cada centro de negócio (direita e esquerda). As comissões são pagas em função do volume movimentado na perna mais fraca, o que acaba estimulando uma corrida entre os grupos. Após atingir o patamar máximo na tabela de remuneração, o distribuidor pode abrir novos centros de negócio, aumentando indefinidamente seu potencial de ganhos [53, 32].

4. No Australiano, o distribuidor (vendedor que convida outras pessoas) não ganha absolutamente nada sobre os dois primeiros patrocinados (convidados) diretos e seus respectivos grupos (convidados de quem ele convidou). Seus rendimentos são calculados com base no terceiro direto em diante, considerando apenas os dois primeiros patrocinados de cada um em suas linhas descendentes [53, 32].

5. A Emancipação Gradativa é organizada em escalas progressivas e emancipações, com objetivo de selecionar os líderes (pessoas que possam treinar outras) com mais critério. Esse plano costuma ser muito seletivo e teve maior destaque nas décadas de 70 e 80 [53, 32]. Um plano semelhante é o *Breakaway/Stairstep*, nele uma pessoa e aqueles a quem ela convidou podem tornar-se independentes enquanto organização, por terem atingido determinado volume de operações (vendas). Neste caso, o patrocinador original continua recebendo *overrides* pelo volume de operações da rede. *Override* é o valor pago sobre a produção (venda) dos integrantes de determinada "perna". Perna é uma linha de distribuidores que inicia com uma pessoa patrocinada e continua abaixo deste distribuidor [53, 81].

2.4.2 Comissão

De um modo geral, promete-se 25% de lucro sobre as vendas. O recrutado também pode cadastrar outros vendedores e receber 10% de comissão sobre todas as vendas que eles fizerem (1° nível), além de mais 5% de lucro sobre o 2° nível (venda de todos os distribuidores patrocinados pelo 1° nível). Assim, sucessivamente, todos esses distribuidores possuem a possibilidade de cadastrar outras pessoas, formando uma "rede" de distribuidores, os quais vão sendo remunerados a partir de uma escala de percentagens previamente estipuladas para cada nível. Portanto, quanto maior for o número de níveis permitido, maior será o potencial de ganhos dos distribuidores. Algumas empresas prometem distribuir até 73% dos lucros, mas, no exemplo acima, os percentuais acumulados não vão muito além dos 45%, se continuarmos a dividir o percentual a cada nível [53].

2.4.3 Lucratividade

A lucratividade é outro fator a ser considerado nesse modelo de negócio. Para os produtos serem considerados viáveis no Marketing de Rede, a margem de lucro deve ser alta. Ele deve cobrir as despesas do plano de compensação, os programas de incentivo e reconhecimento, ofertas especiais, o preço do produto no varejo, a distribuição e despesas eventuais. Geralmente, as empresas relacionam o desembolso do plano de compensação à lucratividade (rentabilidade) do produto [53, 128].

2.4.4 Aplicação em *crowdsourcing*

Nos sistemas de *crowdsourcing* a alta lucratividade poderia inflacionar as tarefas do ponto de vista dos solicitantes ou pulverizar os ganhos dos trabalhadores, visto que eles teriam que dividí-lo com os níveis mais altos da pirâmide. Por isso e por questões legais, não aplicamos o marketing de rede sobre as remunerações. A recompensa financeira é a motivação central na maior parte dos trabalhos sobre mecanismos de incentivo. Não vamos ignorá-la, mas queremos dar ênfase em como recrutar, envolver e reter os participantes. Esperamos atingir esses objetivos aplicando o marketing de rede sobre "pontos" que classificam a reputação dos trabalhadores.

Capítulo 3

Trabalhos relacionados

Existem diversos tipos de *frameworks*, plataformas, arquiteturas e soluções em geral para *crowdsourcing*. Cada uma com objetivos distintos. Esse panorama dificulta as comparações entre as soluções, principalmente quando incluímos a variável da geolocalização. Nesse caso, fica mais fácil encontrar soluções voltadas para *crowd sensing*. Apesar disso, encontramos trabalhos similares à Snow Flurry. Porém, a comparação ficou restrita a análises qualitativas em virtude da indisponibilidade das soluções. Os próprios trabalhos citados aqui se concentram em *surveys*, idealizações não implementadas, apresentações sem métricas, simulações com variações da própria solução e desenvolvimentos matemáticos.

3.1 Escalabilidade

Apresentaremos nessa seção algumas soluções do estado da arte que almejam a escalabilidade, mas que se revelam caras, complexas ou que apresentam pontos na arquitetura que funcionam como gargalos, impedindo que o sistema seja escalável.

3.1.1 Algoritmos

No artigo de To, Shahabi e Kazemi [184], os trabalhadores enviam a geolocalização para um servidor e, posteriormente, esse servidor atribui tarefas aos trabalhadores que estão próximos ao local de interesse. Nesse trabalho, o objetivo foi maximizar o número total de tarefas atribuídas. Os autores definiram formalmente este problema como atribuição máxima de tarefas (MTA, do termo em inglês *maximum task assignment*) em *crowdsourcing* espacial. Eles propuseram soluções alternativas para enfrentar esse desafio, explorando as

propriedades espaciais do problema, incluindo a distribuição espacial e o custo da viagem. MTA é baseado na suposição de que todas as tarefas são do mesmo tipo e que todos os trabalhadores são igualmente qualificados para resolve-las. Segundo os autores, quando a qualificação varia, estamos diante da atribuição de pontuação máxima (MSA, do termo em inglês *maximum score assignment*). A arquitetura Snow Flurry (apresentada no Capítulo 4) não sofre com esses problemas, o Auspice consegue identificar os trabalhadores em determinado espaço e o programa local, ao criar uma notificação, permite a seleção de um grupo determinado, dentro de uma sub-região. O trabalho de Kazemi e Shahabi [106] defende a mesma ideia de To, Shahabi e Kazemi [184]. Primeiro, os trabalhadores enviam suas localizações para um servidor central. Depois, esse servidor atribui as tarefas conforme a localização dos trabalhadores. Essas soluções utilizam o método *Pull*, nele o trabalhador procura pelas tarefas. A Snow Flurry utiliza o *Push*, nesse método a tarefa é enviada ao trabalhador. Segundo Kandappu et al. [102], o *Push* aumenta o número de tarefas concluídas e reduz a sobrecarga. Além disso, a arquitetura Snow Flurry resolve a atribuição das tarefas localmente, no programa local dos solicitantes, semelhante ao proposto por Deng, Shahabi e Zhu [64]. Quando o número de solicitantes cresce o programa local não fica sobrecarregado, pois opera sobre um pequeno conjunto de trabalhadores obtidos do GNS. Ainda na linha das atribuições de tarefas, o artigo de Dang, Nguyen e To [59] apresenta o problema da máxima atribuição de tarefas complexas (MCTA, do termo em inglês *Maximum Complex Task Assignment*). Para os autores, a maioria dos *frameworks* para *crowdsourcing* espaciais assumem tarefas independentes e atômicas. No entanto, pode haver necessidade de uma tarefa espacial complexa, composta por algumas subtarefas. A atribuição de atividades complexas exige atribuições de todas as suas subatividades. As estruturas atualmente disponíveis não são aplicáveis a esse tipo de operação. Nesta tese, consideramos as subtarefas um objetivo técnico para trabalhos futuros.

O objetivo de Deng, Shahabi e Demiryurek [63] é encontrar um arranjo para os trabalhadores que maximize o número de tarefas executadas. Eles provam que o problema é NP-difícil. Depois, apresentam dois algoritmos exatos baseado em programação dinâmica e *branch-and-bound*. Como algoritmos exatos não tratam um grande número de tarefas e/ou estão limitados à quantidade de recursos disponíveis em plataformas móveis, eles também propuseram aproximações e algoritmos progressivos. Entendemos que no método *Push* a possibilidade de recusa das tarefas por parte dos trabalhadores e o prazo para aceite sejam suficientes para tratar esse critério.

O problema da segurança é abordado por To, Ghinita e Shahabi [183]. Os autores afirmam que as soluções atuais exigem dos trabalhadores a divulgação da localização a

entidades não confiáveis. O artigo apresenta uma estrutura que protege a privacidade da geolocalização dos trabalhadores, por meio de privacidade diferencial e *geocasting*. A solução exposta no Capítulo 4 não aborda essa questão, visto que, pelo escopo, as informações sobre geolocalização estão no Auspice. Consideramos cenários onde a exposição desses dados não comprometem a segurança dos usuários. Uma camada de segurança que não interfira na escalabilidade é um tema para trabalhos futuros. Shahabi [165] segue uma linha semelhante a essa, afirmando que dois dos principais impedimentos para o sucesso dos sistemas MCS, em aplicações do mundo real, são problemas de escalabilidade e confiança. Sem considerações de escala, é impossível desenvolver um sistema genérico multicampanha de *crowdsourcing* espacial que possa, de forma eficiente e em tempo real, distribuir tarefas de muitos solicitantes para inúmeros trabalhadores. Sem confiança, o MCS não pode avaliar a credibilidade dos dados fornecidos, tornando-o ineficaz para substituir os meios de coleta de dados tradicionais.

A confiança também foi abordada no trabalho de Kazemi, Shahabi e Chen [107]. Nele, o foco está na qualidade das respostas dos trabalhadores. Para os autores, um dos principais desafios em *crowdsourcing* espacial é como verificar a validade dos resultados fornecidos pelos trabalhadores, quando os mesmos não são confiáveis. Para resolver este problema, eles assumiram que todos têm uma pontuação de reputação que afirma a probabilidade do trabalhador executar uma tarefa corretamente. Além disso, definiu-se um nível de confiança para cada tarefa espacial. Logo, uma resposta à determinada tarefa espacial só é aceita se a confiança for superior a um determinado limite. Dessa forma, eles estão tentando resolver o problema da maximização do número de tarefas espaciais que são atribuídas a um conjunto de trabalhadores, desde que satisfaçam os níveis de confiança dessas tarefas. O aspecto original do problema é que a alocação ótima de tarefas depende fortemente das localizações geográficas dos trabalhadores e das tarefas. Isto significa que cada tarefa espacial deve ser atribuída a um número suficiente de trabalhadores, de tal forma que a sua reputação agregada satisfaça a confiança da tarefa. Por conseguinte, uma abordagem exaustiva precisa calcular a pontuação de reputação agregada usando um mecanismo típico de agregação de fusão de decisão, como o voto, para todos os subconjuntos possíveis de trabalhadores, o que torna o problema NP-difícil. De forma semelhante, a Snow Flurry possui um sistema de reputação que utiliza pontuação de opinião média (MOS, do termo em inglês *Mean Opinion Score*) para avaliar as respostas dos trabalhadores.

Um caso parecido com o trabalho de Kazemi, Shahabi e Chen [107] pode ser visto no artigo de Cheng et al. [48]. Essa pesquisa apresenta o problema da confiança baseada em

diversidade (RDB-SC, do termo em inglês *reliable diversity-based spatial crowdsourcing*). Nele, as tarefas espaciais, tais como capturar vídeos/fotos de um marco ou espetáculos de fogos de artifício e verificar se existem ou não vagas disponíveis no estacionamento, têm restrições de tempo e os trabalhadores estão se movendo em algumas direções. O problema RDB-SC consiste em designar trabalhadores à tarefas espaciais tais que a confiabilidade da conclusão e as diversidades espaciais/temporais das tarefas são maximizadas. Os autores provaram que o problema RDB-SC é NP-difícil e intratável. Logo, eles propuseram três abordagens de aproximação eficazes, incluindo algoritmos gananciosos, de amostragem e dividir para conquistar. A fim de melhorar a eficiência, eles também projetaram um eficaz índice baseado em custo-modelo. Ele pode tratar a dinâmica da movimentação dos trabalhadores e tarefas espaciais com baixo custo. A arquitetura desta tese permite que o programa local do solicitante trabalhe com um pequeno conjunto de trabalhadores selecionados por contexto. Pesquisas sobre a dinâmica das solicitações no MTurk mostraram que as campanhas dificilmente operam com mais de 200 trabalhadores. Logo, os estudos dos problemas inerentes as grandes campanhas fazem mais sentido quando temos no centro um sistema de *crowd sensing*.

O esboço de uma arquitetura é apresentado por Chen et al. [47]. Trata-se de uma plataforma de *crowdsourcing* espacial de aplicação geral chamada gMission. Ela apresenta uma coleção de novas técnicas, incluindo detecção geográfica dos trabalhadores e recomendação de tarefas. A arquitetura Snow Flurry também cobre esses tópicos com escalabilidade.

A maioria dos sistemas de *crowdsourcing* existentes depende de servidores centrais, que estão sujeitos às deficiências do modelo tradicional baseado em confiança, como o ponto único de falha. Eles também são vulneráveis a ataques distribuídos de negação de serviço (DDoS) e Sybil devido ao envolvimento de usuários mal-intencionados. Além disso, altas taxas de serviço da plataforma podem impedir o desenvolvimento do *crowdsourcing*. Para lidar com essas questões Li et al. [115] conceituaram uma estrutura descentralizada baseada em *blockchain* para *crowdsourcing* chamada CrowdBC, na qual a tarefa de um solicitante pode ser resolvida por uma multidão de trabalhadores sem depender de nenhuma terceira instituição confiável, a privacidade dos usuários pode ser garantida e apenas baixas taxas de transação são requeridos. No início das pesquisas desta tese consideramos o uso do *blockchain*, mas encontramos alguns problemas nessa tecnologia. Ela, basicamente, funciona como uma lista simplesmente encadeada sem remoção, devido ao fator que garante a segurança do sistema. Remover um item ou quebrar a lista significa quebrar a segurança. Essa lista não fica em um único servidor na rede, como um sistema

cliente/servidor, mas é replicado em vários e para que uma operação seja considerada válida ela precisa ser confirmada por 51% desses servidores, o que provoca atrasos no processamento das transações. Quanto mais operações são realizadas, maior a lista se torna, exigindo mais espaço em disco e capacidade de processamento para percorre-la em busca de registros. Com o tempo, as operações se tornam cada vez mais lentas e os servidores cada vez mais saturados em sua capacidade de atendimento. O que se observa é a distribuição dos problemas atribuídos ao modelo cliente/servidor, com o agravante de não poder otimizar a base de dados.

3.1.2 Computação em nuvem

O estudo de Alfarrarjeh, Emrich e Shahabi [8] demonstrou a ineficiência do servidor único, quando um grande número de trabalhadores e tarefas chegam num curto espaço de tempo. Ele idealizou a atribuição de tarefas como um grafo bipartido, depois elaborou uma solução que particiona a carga entre um conjunto de servidores em nuvem. Já a pesquisa de Deng, Shahabi e Zhu [64] voltou seus estudos para os algoritmos e alerta que as melhores soluções em atribuição de tarefas são lentas e não escalam. Então, eles desenvolveram uma estrutura que divide recursivamente todos os trabalhadores e tarefas em diferentes partições, permitindo uma atribuição localmente em um espaço muito menor e promissor. A Snow Flurry resolve a atribuição das tarefas localmente, no programa local dos solicitantes, semelhante ao proposto por Deng, Shahabi e Zhu [64]. Quando o número de solicitantes cresce o programa local não fica sobrecarregado, pois opera sobre um pequeno conjunto de trabalhadores obtidos do GNS.

3.1.3 *Proxy*

No estado da arte em escalabilidade para *crowdsourcing*, vemos os artigos que contribuíram com soluções baseadas em *proxy*. Na proposta de Yan et al. [198] é apresentado o mCrowd, uma plataforma de *crowdsourcing* móvel, voltada para iPhone, que funciona como um *proxy* entre os usuários (solicitantes e trabalhadores) e as plataformas Amazon Mechanical Turk e ChaCha. Analisando a proposta, podemos concluir que o mCrowd deve enfrentar um problema parecido com o encontrado no modelo cliente-servidor. Ainda que as plataformas parceiras possam garantir escalabilidade, o mCrowd pode representar um gargalo entre elas e os usuários. Para evitar esse problema, Hara et al. [85] propôs um sistema descentralizado, onde servidores especiais, conhecidos como *Server Access Network Entities* (SANE), são distribuídos geograficamente para diluir a carga. Porém, esse mo-

delo incorre em custo e complexidade. O Auspice dilui a carga para a arquitetura Snow Flurry de forma semelhante, ele possui diversos servidores e posiciona as informações geograficamente próximas dos interessados. O posicionamento das informações é outro ponto forte no Auspice, pois ele utiliza um algoritmo chamado Gigapaxos [187] que tem a capacidade de gerenciar e reconfigurar rapidamente um número muito grande de grupos de réplicas.

3.1.4 Redes tolerantes à atrasos

Outra linha de pesquisa envolve as redes tolerantes a atrasos (DTN, do termo em inglês Delay-Tolerant Networking) que, em resumo, é uma tecnologia capaz de montar uma rede sem infraestrutura. Os trabalhos com essa tecnologia têm o seguinte cenário em comum: precisam reunir informações de um lugar remoto ou área de desastre. Segundo Al Noor e Hasan [7], essa tarefa não é trivial, dada a indisponibilidade de infraestrutura apropriada e devido ao desempenho insatisfatório das abordagens atuais com DTN. Diante disso, os autores desenvolveram um *crowdsourcing* que combina rede celular e uma forma de nuvem tolerante a atraso. A ideia é estabelecer uma forma de comunicação direta entre os dispositivos para auxiliar à transmissão de dados ao longo da rede.

3.1.5 Comunicação direta em redes 5G

O trabalho de Han e Wu [84] trata da comunicação direta entre dispositivos, afirmando que a tecnologia Device-to-Device (D2D) é altamente desejada quando o solicitante não pode alcançar diretamente os trabalhadores ou quando as abordagens convencionais para o transporte de dados têm custo elevado. O D2D faz parte das pesquisas sobre redes 5G que prometem uma comunicação fim-a-fim e velocidades muito superiores as 4G. A Snow Flurry também trabalha com coleta direta, mas, devido à abrangência, optamos por um modelo viável em 3G, 4G e Wi-Fi.

3.1.6 Virtualização e suporte à mobilidade

No artigo de Xiao et al. [196], encontramos um *survey* sobre escalabilidade em sistemas Mobile Crowd Sensing e uma proposta que usa máquinas virtuais com o objetivo de (1) separação da coleta de dados e compartilhamento lógico específico do aplicativo; (2) remoção da instalação do aplicativo para *smartphone* a partir do caminho crítico da implantação do aplicativo; e (3) descentralização do processamento e agregação de dados

próximo à fonte de dados. No entanto, os autores reconhecem o risco de sobrecarga de virtualização e falta de suporte à mobilidade. A Snow Flurry não apresenta sobrecarga e oferece suporte à mobilidade por meio da arquitetura MobilityFirst.

3.1.7 Computação de borda

A computação de borda em sistemas de *crowd sensing* foi tema no artigo de Marjanovic, Antonic e Zarko [120]. Basicamente, esse trabalho defende uma arquitetura em nuvem com servidores intermediários (borda) mais próximos dos usuários para reduzir a latência. A arquitetura Snow Flurry já faz isso, visto que o Auspice conta com servidores espalhados pelo mundo que organizam os dados de forma a deixá-los geograficamente próximos dos usuários. A pesquisa de Marjanovic, Antonic e Zarko [120] foi a que mais se aproximou do tema escalabilidade, porém o trabalho concentrou esforços em descobrir quantos servidores intermediários seriam necessários para viabilizar a proposta deles e se limitaram a afirmar que a computação em nevoa pode proporcionar uma latência ultrabaixa (menos de 1 ms) sem apresentar avaliações. No trabalho de Sharma et al. [167] podemos encontrar todo um estudo e os resultados dos experimentos sobre a latência do Auspice e como ele pode atender rapidamente as atualizações de geolocalização dos trabalhadores e as buscas por trabalhadores feitas pelos solicitantes.

3.1.8 Medusa e a computação elástica

Por fim, devido à indisponibilidade de sistemas Mobile Crowdsourcing, até o limite do nosso conhecimento, esta tese apresenta uma comparação com o sistema Medusa de Ra et al. [145]. O Medusa é uma solução de *crowd sensing* que oferece serviços com alto nível de abstração na especificação das etapas necessárias para concluir uma tarefa de detecção e que emprega um sistema distribuído para coordenar a execução dessas tarefas entre *smartphones* e um *cluster* na nuvem. Os autores também afirmam que os módulos funcionam de forma independente apesar de existir uma orquestração central e que, por isso, a arquitetura poderia fazer uso da computação elástica para obter escalabilidade.

3.2 Travessia de NAT

A solução utilizada na Snow Flurry foi inspirada em ferramentas de mercado como AnyDesk[1] e TeamViewer[2], mas a intenção primária desta tese era estabelecer uma comunicação fim-a-fim com o MSocket da arquitetura MobilityFirst. Porém, a adaptação para Android ainda não foi disponibilizada a tempo pela Universidade de Massachusetts (UMass) e, portanto, esse passa a ser um objetivo para trabalhos futuros. As redes 5G também se preocupam em fornecer meios de comunicação direta entre dispositivos, mas uma das vantagens do MSocket sobre essa tecnologia e as soluções do estado da arte é a capacidade de estabelecer comunicação fim-a-fim, com suporte à mobilidade, sobre os atuais meios de conexão.

3.2.1 Protocolo STUN

As aplicações típicas de NAT resolvem os problemas de comunicação ponto-a-ponto por meio da abordagem transversal NAT como a utilizada pelo protocolo STUN. Por exemplo, o Skype é um aplicativo NAT amplamente conhecido. O desenvolvedor do Skype alega que o serviço de voz sobre IP (VoIP) pode funcionar através de NATs/*firewalls*. Dois clientes do Skype que residem em diferentes NATs podem se conectar diretamente entre si. No entanto, os clientes Skype não podem atravessar o *NAT daemon* no sistema FreeBSD sem ativar a opção "Deny Incoming". O padrão dessa opção é desativado e os clientes do Skype devem depender de terceiros para retransmitir voz em vez de entrar em contato diretamente com o correspondente. A situação resulta em problemas de degradação e escalabilidade da qualidade de voz. Chang et al. [40] propõem uma nova abordagem "KaiKai" para resolver este problema. O KaiKai pode fazer com que dois clientes entrem em contato diretamente, mesmo que esses dois clientes residam em diferentes NATs. Usando a análise de comportamento de protocolo, o KaiKai pode cooperar com NATs com ou sem a opção "Deny Incoming". Consequentemente, a melhor qualidade de serviço e escalabilidade podem ser alcançados.

3.2.2 Protocolo SIP

Além das propostas com STUN, podemos encontrar outras abordagens como a apresentada no artigo de Chen et al. [41] que detalha uma plataforma de comunicação baseada

[1] https://anydesk.com/pt/
[2] https://www.teamviewer.com/pt-br/

no protocolo de iniciação de sessão (SIP, do termo em inglês *Session Initiation Protocol*) e um novo algoritmo de criptografia para garantir a segurança da comunicação. A comunicação com MSocket também utiliza criptografia e serviços mais sensíveis também utilizam criptografia na API HTTP de comunicação com o Auspice.

Algumas soluções emergentes sugerem o uso do SIP para configurar o encapsulamento em UDP, usando o *Universal Plug and Play* (UPnP) ou até mesmo implantando o IPv6. Eppinger [70] argumenta que as abordagens acima sofrem de problemas de escalabilidade, não abordam problemas de mobilidade, exigem implantação de nova infraestrutura de rede ou exigem o uso de interfaces de comunicação, pilhas de comunicação e protocolos de segurança não padronizados. Então, os autores apresentam o NatTrav que faz conexões TCP entre *peers* NAT e aborda todas as preocupações acima.

O protocolo SIP foi proposto para serviços multimídia e conectividade de área ampla em ambientes domésticos inteligentes (SHEs, do termo em inglês *Smart Home Environments*). Um problema importante para a implantação do SIP em SHEs é a passagem dos pacotes pelo conversor de endereços de rede (NAT). Os pacotes SIP e RTP são entregues entre uma rede privada e a Internet através da função NAT de um *gateway* doméstico, onde o NAT traduz a dupla endereço IP e porta na camada de transporte, mantendo a camada de aplicação inalterada. Isso resulta em inconsistência entre os endereços IP/números de porta nas camadas transporte e aqueles da camada SIP. Para resolver esse problema, Chen, Huang e Chao [44] descreveram seis soluções, incluindo rota estática, UPnP, STUN, ICE, ALG e SBC. Eles compararam essas soluções em termos de eletrodomésticos inteligentes (SHA, do termo em inglês *Smart Home Appliance*), escopo de NATs suportados, passagem de NAT multicamada, facilidade de configuração, problemas de segurança e complexidades de tempo. No caso do MSocket, já comentamos que existe a necessidade de reescrever o código Java com o MSocket no lugar do Socket convencional.

O artigo de Gaylani e Erten [79] aborda a travessia de NAT e situações onde um dos servidores se move durante uma sessão ativa para um novo local que também está atrás de um NAT. Algumas soluções são apresentadas, com base nas mensagens de IP móveis e SIP. As soluções são baseadas em perfuração e na suposição de que os servidores já conhecem os detalhes um do outro. Portanto, não se faz necessário consultar os servidores de *proxy* ou de localização ao restabelecer a comunicação, reduzindo o tempo de transferência necessário.

3.2.3 Outras abordagem e atravessando *firewalls*

Nos artigos de Tanaka et al. [177, 178] vemos trabalhos focados no acesso remoto a dispositivos ECHONET Lite. Eles propuseram o *Network Traversal with Mobility* (NTMobile), que pode resolver o problema da passagem de NAT, obter comunicação criptografada de ponta a ponta e mobilidade IP. O principal problema dessa proposta é sua especialização, se comparada ao MSocket.

Chen e Jia [46] apresentaram uma proposta de resolução de nomes que consiste num *Bi-directional NAT* (BNAT) e um *Domain Name System Application Level Gateway* (DNS_ALG). Esses componentes coordenam e fornecem capacidade de acesso bidirecional entre Intranet e Internet, onde todos os servidores privados podem ser endereçados por um nome de domínio totalmente qualificado. A ideia é parecida com a GUID do GNS, mas o trabalho envolve estações sem mobilidade.

O artigo de Guha e Francis [83] destaca que apenas atravessar o NAT não é suficiente. Mesmo onde existem endereços globais, os *firewalls* não conseguem obter informações suficientes sobre um fluxo de cabeçalhos de pacotes e, com frequência, erram, geralmente por serem excessivamente conservadores: não permitindo fluxos que poderiam ser permitidos de outra forma. Os autores apresentam uma nova arquitetura, um projeto de protocolo e uma implementação para o estabelecimento de fluxo na Internet. A arquitetura, chamada NUTSS, leva em consideração as políticas combinadas de terminais e provedores de rede. A NUTSS não requer alterações nos protocolos de rede existentes e, combinada com as técnicas recentes de NAT, funciona com IPv4 e *firewall*/NAT existentes. Uma análise do comparativa entre a NUTSS e a MobilityFirst, sob a ótica dos *firewalls* é um tema para trabalhos futuros visto que o NUTSS alega prover *multi-homing* e mobilidade.

3.2.4 Conexão por TCP

Na imagem atual da Internet, mais de 70% dos servidores estão localizados atrás de NATs [56]. Isso não é um problema para o paradigma cliente/servidor. No entanto, a Internet evoluiu e, atualmente, a maior parte do tráfego é devida a aplicativos P2P. Esse cenário apresenta um desafio importante: dois servidores por trás de NATs não podem estabelecer comunicações diretas. A maneira mais fácil de resolver esse problema é usar uma terceira entidade, chamada *Relay*, que encaminha o tráfego entre os servidores. Embora muitos esforços tenham sido dedicados para evitar o uso de *Relays*, eles ainda são necessários em muitas situações. Assim, a seleção de um *Relay* adequado torna-se

crítica para muitas aplicações P2P. Cuevas et al. [56] propõem um *Gradual Proximity Algorithm* (GPA): Um algoritmo simples que garante a seleção de um *Relay* topologicamente próximo. Eles apresentaram uma análise baseada em medições, mostrando que o GPA minimiza tanto o atraso da comunicação retransmitida quanto o tráfego gerado pelo *Relay*, sendo uma solução de QoS-aware e outra de ISP-friendly. Além disso, o artigo apresenta a *Peer-to-Peer NAT Traversal Architecture* (P2P-NTA), que é uma solução global, distribuída e colaborativa, baseada no GPA. Esperamos obter uma transmissão sem intermediários com o MSocket, sem atrasos nas comunicações e o tráfego extra gerado pelo GPA.

Uma proposta que não utiliza servidores intermediários é apresentada por Liu et al. [116], onde os autores apresentam uma abordagem com TCP reverso. A implementação apresentada nesta tese utiliza UDP, mas a solução futura, ao utilizar o MSocket, vai operar com o protocolo TCP. O TCP é preferível em muitos casos devido ao modelo de garantia na entrega, mas seu uso em travessia de NAT não é trivial. Nos últimos anos, a comunidade de padrões desenvolveu técnicas para atravessar caixas de NAT/*firewall* com UDP. Devido à natureza assimétrica do estabelecimento de conexões TCP, no entanto, a passagem NAT com TCP é mais difícil. Pesquisadores propuseram recentemente uma variedade de abordagens promissoras para o TCP NAT transversal. O sucesso dessas abordagens, no entanto, depende de como as caixas NAT respondem a várias sequências de pacotes TCP (e ICMP). O artigo de Guha e Francis [82] apresenta o primeiro estudo amplo do comportamento de NAT para um conjunto abrangente de técnicas de travessia de NAT com TCP em uma ampla gama de produtos comerciais da NAT. Eles desenvolveram um conjunto de testes de *software* publicamente disponível que mede as respostas dos NATs para uma variedade de sondagens isoladas e para concluir os estabelecimentos de conexão TCP.

Na pesquisa de Liu et al. [117], vemos a proposta do TCPBridge. O TCPBridge converte o TCP traversal para o percurso UDP sem modificar nenhum binário dos aplicativos baseados em TCP. Esse *design* pode ser integrado com as aplicações P2P que não resolveram o problema de travessia com TCP, estendendo-as para suportar comunicações diretas entre os servidores atrás dos NATs. Os autores também argumentam que o TCPBridge é escalável e robusto.

D'Angelo e Rampone [60] descrevem uma nova arquitetura de *Video Surveillance as a Service* (VSaaS). A solução proposta usa um componente complementar, chamado WS-Gateway (*gateway* baseado em WebSocket), instalado na rede privada (junto com a

rede de câmeras IP). O protocolo WebSocket é usado para estabelecer uma comunicação bidirecional entre os atores no sistema. A principal vantagem da solução é a superação de problemas de alcance causados pela presença de NATs ou *firewalls* na rede. Em relação ao MSocket, podemos dizer que o VSaaS é muito específico e o trabalho não deixa claro se há suporte à mobilidade.

3.2.5 Utilizando SDN

Muitas soluções NAT transversais, como os utilitários STUN, UPnP, TURN, ALG e ICE, foram propostos. Entre as soluções NAT, o *3rd Generation Partnership Project* adota a solução ALG para serviços multimídia SIP/IMS nas redes móveis. A solução ALG não requer modificações nos clientes, por exemplo, no dispositivo do usuário. No entanto, a solução ALG utiliza um *proxy* para converter os pacotes RTP e, portanto, ela produz atraso extra e possibilidade de perda de pacotes RTP. No trabalho de Chen e Chen [43], os autores integraram uma SDN às redes móveis All-IP e propuseram uma solução para melhorar a solução ALG nas redes móveis All-IP habilitadas para SDN, analisando a mesma em termos de taxa de transferência, taxa de perda e atraso.

3.2.6 Internet do futuro

Para prover tráfego HTTP a partir de dispositivos domésticos através de *gateways*, Raza, Ahmed e Boutaba [149] utilizaram acesso remoto usando DNS de nomes globais, apesar dos dispositivos finais possuírem IPs locais. Na prova de conceito, eles utilizaram a plataforma OpenWRT como base da arquitetura proposta e relataram resultados experimentais de desempenho no acesso aos dispositivos. Essa ideia também se aproxima da proposta desta tese e um outro trabalho interessante é o de Ambrosin et al. [14] que compara as quatro soluções mais promissoras nesta área e que tem as pesquisas financiadas pela *National Science Foundation* (NSF) dos EUA: o *nebula*, *named-data networking*, MobilityFirst e *expressive Internet architecture*. A NSF tem sido um dos principais apoiadores dos esforços para projetar um conjunto de arquiteturas candidatas de Internet de próxima geração. Como um requisito proeminente de *design*, a NSF enfatizou "segurança e privacidade por *design*" a fim de evitar o longo e infeliz histórico de *patching* e *retrofitting* incremental que caracteriza a arquitetura atual da Internet. Ambrosin et al. [14] fornecem uma análise abrangente e neutra de recursos de segurança e privacidade relevantes nessas futuras arquiteturas de Internet financiadas pela NSF. Pesquisas anteriores sobre futuras arquiteturas da Internet fornecem uma comparação limitada, ou mesmo nenhuma, sobre

os recursos de segurança e privacidade. Além disso, este documento também compara os quatro designs candidatos com a arquitetura atual baseada em IP e discute semelhanças, diferenças e possíveis melhorias.

3.3 Envolvimento e qualidade

O mecanismo de incentivo MktPoints engloba diversos conceitos que podem aparecer na literatura sob temas diferentes. Normalmente, encontramos esses elementos em estudos sobre mecanismos de incentivo e sistemas de reputação. Portanto, listamos a seguir o estado da arte sobre cada um.

3.3.1 Mecanismos de incentivo

Algumas propriedades desejadas nos mecanismos de incentivo são: (1) eficiência computacional; (2) racionalidade individual; (3) maximização da utilidade; (4) bem-estar social; e a (5) verdade/confiança. Veremos a seguir como o estado da arte tratou essas propriedades em cada caso.

3.3.1.1 Maximização da utilidade, verdade e a qualidade

No levantamento feito por Zhang et al. [208], vemos a afirmação de que nenhum dos trabalhos, desenvolvidos até o momento, considera ocorrências de natureza oportunista, na área de interesse dos usuários. Especificamente, para uma aplicação geral de detecção de *smartphones*, a plataforma deve distribuir tarefas para cada usuário em sua chegada e tomar uma decisão imediata de acordo com a resposta deles. Para acomodar essa configuração geral, os autores projetaram três mecanismos de incentivo em linha, chamados TBA, TOIM e TOIMAD, com base em leilão reverso *online*. TBA é uma plataforma projetada para perseguir a maximização da utilidade, enquanto TOIM e TOIM-AD buscam a propriedade crucial da verdade (*truthfulness*). Todos os mecanismos possuem as propriedades desejadas de eficiência computacional, racionalidade individual e rentabilidade. Além disso, eles são altamente competitivos em comparação com as soluções ótimas *offline*. A eficiência computacional do mecanismo proposto nesta tese pode ser visto na avaliação da latência que será posteriormente apresentada. Racionalidade individual é a condição em que um trabalhador recebe menos do que poderia obter por conta própria, sem cooperar com qualquer outra pessoa. Nesse aspecto, o MktPoints tem um forte apelo

à formação de grupos como diferencial competitivo, visto que indiretamente pode proporcionar um aumento das tarefas atribuídas e, por consequência, aumento da rentabilidade.

Diferente dos trabalhos existentes, Jin et al. [96] incorporou uma métrica fundamental, chamada de qualidade da informação (QoI, do termo em inglês *Quality of Information*), ao mecanismo de incentivo para sistemas de *Mobile Crowd Sensing* (MCS). Devido a vários fatores (por exemplo, a qualidade do sensor, ruído, etc), a qualidade dos dados sensoriais varia significativamente entre os trabalhadores. A obtenção de dados com alta qualidade e baixo custo é uma busca constante nas plataformas MCS. Tecnicamente, o trabalho apresenta um mecanismo de incentivo baseado em leilões reversos compostos. Os autores investigaram os modelos de leilão combinatórios *single-minded* e *multi-minded*. Para o primeiro, eles projetaram um mecanismo que maximiza o bem-estar social, é individualmente racional, confiável e eficiente computacionalmente. Para o outro, eles projetaram um mecanismo iterativo descendente que chega próximo do ótimo bem-estar social, enquanto satisfaz a racionalidade individual e a eficiência computacional. Entendemos que o mecanismo proposto aqui atinge o bem-estar social por abordar tanto incentivos intrínsecos quanto extrínsecos. Também utilizamos gamificação no MktPoints, mas para proporcionar incentivo sem privilegiar uma das partes.

3.3.1.2 Racionalidade individual e o bem-estar social

Koutsopoulos [110] desenvolveu um mecanismo que determina um nível de participação versus alocação de pagamento que é mais interessante para o solicitante, ou seja, ele minimiza o custo total de compensação aos participantes, ao entregar uma certa qualidade de experiência para os solicitantes de tarefas. Ele tratou o problema no contexto do leilão reverso ideal e mostrou como diferentes qualidades, nas informações apresentada pelos participantes, podem ser rastreadas pela retaguarda e utilizadas nos procedimentos de nível de participação e seleção para pagamento. O resultado foi um mecanismo que resolve de forma ideal o problema acima, e, ao mesmo tempo, é individualmente racional e compatível com incentivo. Esse trabalho apresenta uma solução parecida com a proposta nesta tese, porém com um custo computacional elevado sobre a retaguarda e não incentiva o recrutamento de outros participantes.

O artigo de Lee e Hoh [112] estuda o modelo econômico de incentivo a participação em aplicações de *participatory sensing*. Para estimular a participação do usuário, os autores desenvolveram e avaliaram um mecanismo de incentivo com precificação dinâmica baseada num novo leilão reverso, onde os usuários podem definir um preço e vender os seus

dados de sensoriamento ao provedor de serviço. O mecanismo de incentivo proposto se concentra em minimizar e estabilizar o custo do incentivo, mantendo um nível adequado de participantes, por prevenir que os usuários abandonem as aplicações de *participatory sensing*. Comparado ao mecanismo de seleção aleatória com preço fixo, a proposta não só reduz o custo de incentivo para reter o mesmo número de participantes, mas também melhora a equidade da distribuição de incentivo e o bem-estar social. Também ajuda a alcançar um equilíbrio geográfico nas medições e, mais importante, pode remover o fardo da decisão do preço pelos dados que é o passo mais difícil na criação de mecanismo de incentivo. A solução apresentada no Capítulo 4, não trabalha com a hipótese da prévia coleta de dados dos sensores para posterior venda. Pretendemos utilizar uma prévia definição de preço por unidade de participação, semelhante ao relatado por Lee e Hoh [112]. Logo, esperamos que as ofertas por nível de reputação regulem os preços e reduzam o fardo da precificação.

3.3.1.3 Eficiência computacional

Segundo Zhao, Li e Ma [210], os mecanismos existentes atendem cenários *offline*. As informações dos usuários são conhecidas *a priori*. Diante disso, eles buscaram algo mais realista. Consideraram que os usuários chegam um por um, de forma *online* e aleatória. Utilizaram leilão *online* e investigaram o envio de dados particulares, por parte dos usuários, ao *crowdsourcer* quando chegam. O sistema têm o objetivo de selecionar um subconjunto desses usuários antes de um prazo especificado e maximizar o valor dos serviços fornecidos sob uma restrição orçamentária. O trabalho apresenta dois mecanismos *online*, OMZ e OMG, satisfazendo a eficiência computacional, a racionalidade individual, viabilidade orçamentária, a verdade (*truthfulness*) e a soberania do consumidor. O MktPoints trabalha com informações georreferenciadas em tempo real, com orçamento definido pelo usuário e, como citado anteriormente, ele provê racionalidade individual, eficiência computacional e podemos considerar que as penalidades aplicadas, as tarefas avaliadas negativamente, ajudam na preservação da verdade.

Diferente dos trabalhos anteriores, Feng et al. [75] levaram em consideração a localização do usuário no momento da atribuição das tarefas. Os pesquisadores afirmam que o reconhecimento do local aumenta em grande parte a complexidade teórica e computacional. Para tratar a questão, eles desenvolveram um *framework* de leilão reverso para modelar as interações entre a plataforma e os *smartphones*. Os autores também demonstram que determinar as propostas vencedoras é NP-difícil. Apesar disso, conseguiram

criar um mecanismo chamado TRAC que consiste em dois componentes principais. O primeiro componente é um algoritmo de quase ótima aproximação para determinar as licitações vencedoras com complexidade polinomial, o qual se aproxima da solução ideal dentro de um fator 1 + ln (n), onde n é o número máximo de tarefas sensoriais que um aparelho pode acomodar. O segundo componente é um esquema de pagamento crítico que, apesar da aproximação em determinar propostas vencedoras, garante que a apresentação das propostas dos *smartphones* reflitam os custos reais para execução das tarefas de sensoriamento. Por meio de análises teóricas, extensas e rígidas simulações, os autores demonstraram que o mecanismo proposto atinge verdade (*truthfulness*), racionalidade individual e alta eficiência computacional. Essa proposta envolve um cenário com quase nenhuma interação com o trabalhador, funcionando dentro de um escopo definido pelos custos de sensoriamento. Sua adaptação para custear tarefas que dependam dos humanos (cognitivas) não parece trivial. Estamos apresentando mecanismos de incentivo para *crowd sensing* devido às similaridades com os sistemas de *crowdsourcing* e pela quantidade reduzida de propostas nessa linha de pesquisa.

3.3.1.4 Arrependimento

A abordagem de minimização do arrependimento (*regret*), existente na aprendizagem *online*, foi utilizada no mecanismo de Singla e Krause [172]. Essa abordagem calcula a diferença entre a melhor utilidade que poderia ter sido e a utilidade real obtida, com isso os autores exploraram uma ligação entre leilões de aquisição e o problema *multi-armed bandits* onde um conjunto fixo e limitado de recursos deve ser alocado entre escolhas concorrentes (alternativas) de forma a maximizar seu ganho esperado. A pesquisa resultou numa solução com as seguintes características: (1) orçamento viável, (2) utilidade quase ideal para o solicitante, (3) incentivos compatíveis (realistas) para os trabalhadores e (4) suposições mínimas sobre a distribuição dos custos reais aos trabalhadores. A principal contribuição é um novo mecanismo *no-regret* de fixação de preços e o BP-UCB, esse último realiza aquisições por orçamento em ambientes *online* estocásticos. Em comparação com o estado da arte, obteve-se um aumento de 180% na utilidade. Entendemos que este trabalho aborda a retribuição financeira aos trabalhadores como único incentivo válido e que seu objetivo é reduzi-lo ao máximo. Logo, ele não apresenta uma preocupação com a qualidade da entrega e não parece ter a intenção de incentivar a participação dos trabalhadores, que é um problema de fato, mas a participação de solicitantes que desejam pagar pouco.

Em resposta à escassez e oscilações na rede de energia elétrica, Jain, Narayanaswany e Narahari [94] propuseram o MAB-MDR. Esse mecanismo faz ofertas monetárias à consumidores estratégicos para incentivar a redução do consumo. O trabalho é inspirado num novo mecanismo de incentivo utilizado em *crowdsourcing*. A proposta incorpora características realistas do problema de resposta à demanda, incluindo o tempo variável e função de custo quadrático. Ele utiliza leilões *online*, é compatível com incentivos de estratégia dominante, individualmente racional e consegue minimizar o arrependimento (*regret*) de forma sublinear. Apesar de envolver *crowdsourcing* essa proposta não considera alocações dinâmicas no tempo e espaço, mas reconhecemos que um estudo sobre o arrependimento pode ser alvo de trabalhos futuros.

3.3.1.5 Redução de custos

Para Jaimes, Vergara-Laurens e Labrador [92], nenhum mecanismo de incentivo utiliza informações sobre localização, orçamento definido e restrições de atuação, pontos que tornam o sistema mais realista e eficiente. Eles propuseram um mecanismo de leilão reverso com algoritmo guloso que seleciona um subconjunto representativo dos usuários de acordo com sua localização dado um orçamento fixo. Em comparação com os mecanismos existentes, esse sistema melhora a área coberta em mais de 60%, adquirindo um conjunto mais representativo de amostras após cada rodada, mantendo o mesmo número de usuários ativos no sistema e gastando o mesmo orçamento. O MktPoints utiliza informações sobre localização, orçamento definido e restrições de atuação. Porém, esses dados não tem relação direta com o mecanismo de incentivo. Mesmo assim, conseguimos atender os pontos levantados por Jaimes, Vergara-Laurens e Labrador [92].

Singer e Mittal [170] apresentaram um *framework* para a concepção de mecanismos com garantias em sistemas de *crowdsourcing* remunerados. O *framework* permite automatizar o processo de fixação de preços e atribuição de tarefas para os solicitantes. Ele atua em mercados complexos como o Mechanical Turk da Amazon, onde os trabalhadores chegam de forma *online*. A pesquisa considera que os solicitantes enfrentam restrições orçamentárias e prazos para conclusão das tarefas. Os autores apresentaram um mecanismos para maximizar o número de tarefas sob um orçamento e minimizar os pagamentos dado um número fixo de tarefas a serem concluídas. A solução foi testada na prática, isso possibilitou um estudo sobre o comportamento estratégico dos trabalhadores. A implementação da arquitetura Snow Flurry, por ser um sistema de *mobile crowdsourcing*, já adiciona restrições de tempo e espaço ao processo de seleção dos trabalhadores e ainda

permite o filtro por contexto, o que pode reduzir consideravelmente o número de candidatos elegíveis para a realização das tarefas. Portanto, não consideramos a automação do processo de fixação de preços nesse momento, ficando seu estudo de relevância para trabalhos futuros.

Para resolver o desafio da precificação e do controle da qualidade em *crowdsourcing*, Kamar e Horvitz [100] elaboraram um mecanismo de incentivos que promove o relato verdadeiro (*truthfulness*) e desencoraja a manipulação por parte de trabalhadores e proprietários de tarefas sem a introdução de uma sobrecarga adicional de trabalho. Segundo os autores, o acesso programático ao talento humano através de plataformas de *crowdsourcing* esbarra no problema da especificação de incentivos e no controle da qualidade das contribuições. Metodologias para verificação da qualidade incluem o fornecimento de um pagamento se o trabalho for aprovado pelo proprietário da tarefa e contratação de trabalhadores suplementares para avaliar o trabalho dos contribuintes. Ambas as abordagens colocam um fardo sobre as pessoas e sobre as organizações de comissionamento das tarefas. Pior que isso, elas podem ser suscetíveis à manipulação por parte dos trabalhadores e proprietários de tarefas. Além disso, nem o proprietário da tarefa nem o mercado pode conhecer a tarefa bem o suficiente para ser capaz de avaliar os relatórios de trabalho. Metodologias para incentivar trabalhadores sem verificação externa de qualidade incluem recompensas baseadas no acordo com pares do trabalhador ou com a saída final do sistema. Essas abordagens são vulneráveis a manipulações estratégicas por parte dos trabalhadores. Experiências recentes no Mechanical Turk têm demonstrado a influência negativa de manipulações por parte dos trabalhadores e solicitantes de tarefas em sistemas de *crowdsourcing*. Mesmo com o relatado por Kamar e Horvitz [100], acreditamos que o MktPoints pode promover valores adequados para ambas as partes e ajustáveis segundo as forças de oferta e demanda. Além disso, como o processo ocorre de forma menos dinâmica que os leilões, nenhuma das partes tendem a estratégias oportunistas vinculadas ao tempo para coleta das propostas. O leilão é um processo válido para eventos esporádicos, mas não parece ser o mais adequado para eventos frequentes. Por exemplo, considere um cenário onde milhões de tarefas são disparadas diariamente ao redor do mundo. Imagine bilhões de trabalhadores ofertando seus lances e verificando de tempos em tempos a existência de novas demandas. Note a sobrecarga sobre todo o sistema e o tempo gasto pelos trabalhadores em atividades que não lhe garantem a atribuição de uma tarefa. A arquitetura Snow Flurry permite filtros por geolocalização e contexto, isso reduz o número de trabalhadores que interessam ao solicitante de uma tarefa. Isso resolve parte do problema,

mas, se utilizarmos o leilão, o trabalhador continuará investindo parte do tempo na busca por tarefas e na disputa dos leilões.

Koutsopoulos [110] abordou o problema dos mecanismos de incentivo a contribuintes de dados em aplicações de *participatory sensing*. Nesse contexto, um servidor recebe tarefas referentes à área de interesse do solicitante e inicia um leilão para participação dos trabalhadores. À pedido, cada trabalhador relata seu custo percebido em unidades de participação que essencialmente mapeia o montante solicitado de compensação para colaboração. O custo de participação quantifica a insatisfação provocada nos usuários relativa à participação. Este custo é considerado informação privada para cada dispositivo, uma vez que depende fortemente de vários fatores inerentes a ele, como o custo de energia para detecção, processamento e transmissão de dados para o ponto mais próximo de acesso sem fio, o nível da bateria residual, o número de empregos simultâneos no processador do dispositivo, a largura de banda necessária para transmitir dados e os encargos sociais do operador de rede móvel, ou mesmo o desconforto do usuário devido ao esforço manual para enviar dados. Entretanto, os próprios autores reconhecem que, nesse cenário, os participantes tendem a super valorar o custo, ou seja, declarar um custo mais elevado que o real, de modo a obter uma maior retribuição financeira pelo trabalho realizado. Por problemas como esse, optamos por um processo que não utiliza leilão, mas que considera a dinâmica de oferta e demanda ao relacionar os valores cobrados a reputação de cada trabalhador. Dessa forma, o solicitante pode escolher do "nível ideal" de reputação versus valor para cada tarefa designada.

3.3.1.6 Recompensas variáveis

Segundo Neyman [135], o proprietário médio de *smartphones* verifica seu telefone mais de 150 vezes por dia. A partir de 2015, 62% dos usuários de *smartphones* usaram o telefone para procurar informações sobre uma condição de saúde, enquanto 57% usaram o telefone para fazer serviços bancários *online*. Plataformas móveis tornaram-se o meio dominante de interação humano-computador. Então, como esses dispositivos se estabeleceram como nossa conexão com a Internet? A resposta está no *design* viciante. Os *designers* de *software* se tornaram bem versados na criação de *software* que nos cativa em um nível primitivo. Neyman [135] pesquisou estratégias de *design* de *software* viciantes, suas bases em psicologia e suas aplicações em produtos de *software* populares. Como resultado eles oferecem uma nova taxonomia para categorizar melhor essas estratégias de *design* viciante. Dentre as estratégias destacamos aqui o método das recompensas variáveis.

Nesta tese vamos considerar recompensa como algo dado ou recebido em troca por serviço, mérito, dificuldade, etc. O cérebro responde positivamente às recompensas. No *survey* de Neyman [135], podemos ver que as recompensas se tornam recompensas variáveis quando são dadas aleatoriamente e imprevisivelmente. Recompensas variáveis produzem mais do neurotransmissor dopamina do que recompensas regulares. Fora do *design* de *software*, o método de recompensas variáveis intermitentes é usado de forma mais proeminente pelas máquinas caça-níqueis. No entanto, os aplicativos móveis começaram a aproveitar esse efeito por meio da utilização de notificações e outros processos. Ao intensificar os surtos de dopamina recebidos por seus usuários, os projetistas de *software* estão tornando seus produtos viciantes [30].

A psicologia das recompensas tem sido estudada extensivamente, especialmente com relação ao neurotransmissor dopamina e o método das recompensas variáveis tem como base um estudo psicológico que ficou conhecido como "A Caixa Skinner". Esse foi o estudo psicológico mais estreitamente vinculado às recompensas variáveis, na qual pombos e ratos foram condicionados a puxar uma alavanca quando solicitados por uma luz. Os pesquisadores descobriram que os níveis de dopamina nesses pombos e ratos aumentavam quando esperavam uma recompensa. Esses efeitos foram multiplicados quando as guloseimas foram recompensadas aleatoriamente; adicionar variabilidade aumentou a frequência dos pombos completando a ação pretendida [72].

3.3.1.7 Gamificação

O trabalho de Yang et al. [199] apresenta dois modelos de incentivo: um modelo centrado na plataforma que oferece uma recompensa compartilhada aos usuários participantes e outro centrado no usuário que oferece a esse mais controle sobre o pagamento que irá receber. O modelo centrado na plataforma foi desenvolvido usando um jogo de Stackelberg, onde a plataforma é o líder, enquanto os usuários são os seguidores. O artigo mostra como calcular o equilíbrio de Stackelberg, maximizando a utilidade da plataforma sem que os utilizadores possam melhorar sua utilidade unilateralmente, desviando a estratégia corrente. Para o modelo centrado no usuário, eles utilizaram um mecanismo de incentivo baseado em leilão, que é computacionalmente eficiente, individualmente racional, rentável e assertivo.

Fornecer incentivos aos trabalhadores usando um novo modelo da teoria dos jogos, baseado em jogos repetidos. Essa é a proposta de Zhang e Schaar [209]. Para eles, há sempre uma lacuna no bem-estar social, entre o emergente equilíbrio não cooperativo,

quando os trabalhadores perseguem seus próprios interesses, e o resultado eficiente de Pareto. Logo, os autores desenvolveram uma nova classe de protocolos de incentivo com base em normas sociais que integram os mecanismos de reputação ao já existente esquemas de preços, atualmente implementado em sites de *crowdsourcing*, a fim de melhorar o desempenho dos emergentes equilíbrios não-cooperativos em tais aplicações. Em primeiro lugar, formularam as trocas em um *website crowdsourcing* como um mercado de dois lados, onde os solicitantes e os trabalhadores são combinados e jogam *gift-giving* repetidamente. Posteriormente, eles estudaram uma forma de projetar um protocolo ótimo e sustentável (equilíbrio), que atinja o maior bem-estar social no *website*. Eles provaram que o protocolo proposto consegue funcionar perto da eficiência de Pareto. O mecanismo de incentivo defendido no Capítulo 4 também trabalha vinculado a um sistema de reputação. Juntos, permitem que o solicitante direcione tarefas com maior retribuição financeira a trabalhadores com maior reputação.

3.3.1.8 Marketing de rede

Lv e Moscibroda [119] estudaram as árvores de incentivo para motivar a participação de pessoas em sistemas de *crowdsourcing*. Em uma árvore de incentivo, cada participante é recompensado por contribuir com o sistema, bem como por solicitar novos participantes ao sistema, que então contribuem para ele e/ou solicitam novos participantes. Um mecanismo de árvore de incentivo é um algoritmo que determina quanto cada participante individual recebe com base em todas as contribuições dos participantes, bem como a estrutura da árvore de solicitações. A soma das recompensas pagas pelo mecanismo a todos os participantes é linear na soma de sua contribuição total.

As árvores de incentivo têm sido amplamente utilizadas em uma variedade de domínios e sob diferentes nomes, por exemplo, árvores de referência, esquemas de marketing multinível, marketing afiliado ou até mesmo na forma dos infames esquemas ilegais de pirâmide. A questão de como as pessoas podem ser incentivadas, usando esses modelos, para participar de sistemas de *crowdsourcing* passou a ser de interesse significativo para a comunidade de pesquisa. Principalmente, após o trabalho nas *Lottery Trees* de Douceur e Moscibroda [66], e mais proeminentemente através do trabalho da equipe do MIT no "Desafio do Balão Vermelho" [140].

O mecanismo proposto por Lv e Moscibroda [119] não foi implementado. O foco principal do trabalho foram as provas matemáticas. No entanto, se fosse implementado,

ele traria consigo o problema de ser legalmente inviável, por aplicar o marketing de rede sobre a receita da plataforma.

3.3.1.9 Considerações

Diante do que foi apresentado, acreditamos que o MktPoints é a solução mais indicada em cenários com maior frequência de eventos, com restrições de data, hora e local. Nesse sistema, os trabalhadores e solicitantes estipulam seus valores, sem a necessidade de um acompanhamento constante. O leilão tem a vantagem de só atribuir tarefas a trabalhadores que estejam disponíveis. Caso alguém se encontre sobrecarregado, basta não participar de novos leilões. Porém, o suporte à contexto do Auspice nos permite criar um que represente a situação do trabalhador. Por exemplo, quando um trabalhador mudar sua situação para "indisponível", ele não aparecerá para o solicitante. Logo, isso evita atribuições erradas e sobrecarga.

3.3.2 Sistemas de reputação

Por um lado, as plataformas MCS precisão de mecanismos de incentivo para atrair e manter trabalhadores engajados. Por outro, os sistemas de reputação são requeridos para auxiliar os solicitantes no processo de seleção dos trabalhadores. A seguir vamos apresentar alguns pontos desse processo que formam destacados por trabalhos do estado da arte.

3.3.2.1 Confiança versus qualidade

O *crowdsourcing* é útil para utilizar a inteligência e habilidade das pessoas. No entanto, como em qualquer sistema aberto que envolve a troca de coisas de valor, existem comportamentos egoístas e maliciosos que precisam ser mitigados. O gerenciamento da confiança é uma solução viável em muitos sistemas, mas existe uma grande diferença entre os sistemas de *crowdsourcing* e modelos de confiança existentes concebidos para sistemas multiagente. Os curadores humanos têm uma capacidade limitada para tratar tarefas por unidade de tempo, se comparados aos programas de agentes inteligentes. O artigo de Yu et al. [205] destaca um problema nos atuais métodos de tomada de decisão baseados na verdade (*trust-aware*) para atribuição de tarefas em plataformas de *crowdsourcing*. De um lado, os métodos baseados na confiança sobrecarregam os trabalhadores confiáveis. Do outro, as soluções baseadas em carga de trabalho não dão garantias suficientes

sobre a qualidade do trabalho. A solução proposta por eles (SWORD), estabelece um equilíbrio entre esses dois pontos. Os autores também mostraram, através de simulações abrangentes, que o SWORD promove uma melhora significativa no bem-estar social. Reconhecemos o valor dessa proposta quando estamos diante de um cenário com milhares de trabalhadores disponíveis e só precisamos de alguns, mas o funcionamento da Snow Flurry reduz consideravelmente o número de trabalhadores elegíveis e, mesmo em áreas densas, ainda podemos aplicar filtros por contexto para refinar a escolha, dispensando esse tipo de preocupação.

3.3.2.2 Seleção por credibilidade calculada

A seleção dos trabalhadores é uma questão importante e desafiadora em sistemas de *crowdsourcing*. Essa seleção é geralmente baseada em uma avaliação da reputação individual dos trabalhadores que participam em tais sistemas. No entanto, a avaliação da credibilidade e o calculado que adequá tal reputação é um verdadeiro desafio. No trabalho de Allahbakhsh et al. [12], foi proposto um modelo de gestão da reputação que aproveita os valores das tarefas concluídas, a credibilidade dos avaliadores e o tempo de avaliação dos resultados. O objetivo é calcular métricas de qualidade mais confiáveis para os trabalhadores e avaliadores. O modelo foi implementado e validado experimentalmente. Consideramos o cálculo da reputação um fator importante para a proposta desta tese, por um lado queremos representar a reputação mas por outro queremos que esse valor funcione como um gatilho de "recompensa variável".

Um grande desafio no *crowdsourcing* espacial é melhorar a qualidade das respostas. Investigar o impacto das limitações orçamentarias na qualidade das respostas foi o foco de Yu et al. [204]. Para os autores, a maior parte das pesquisas têm se concentrado em trabalho voluntário. Logo, diversos pontos não foram abordados. Neste trabalho, por exemplo, eles levaram em consideração casos realistas, como os pagamentos desiguais com base na confiabilidade. O trabalho propõe uma nova abordagem qualidade/orçamento na alocação de tarefas espaciais. Ela considera conjuntamente reputação e proximidade dos trabalhadores aos locais das tarefas para maximizar a qualidade esperada dos resultados, enquanto permanece dentro de um orçamento limitado. O MktPoints também resolve essa questão, permitindo que o solicitante contrate com base em determinado nível de reputação.

3.3.2.3 Penalidades

A contribuição chave de Jagabathula, Subramanian e Venkataraman [91] é a concepção de um algoritmo de reputação eficiente computacionalmente que identifica e filtra trabalhadores em sistemas de *crowdsourcing*. O algoritmo usa o conceito de ótimas *semi-matchings* em conjunto com penalidades baseadas em divergências nos rótulos. Dessa forma, ele atribui uma pontuação de reputação a cada trabalhador. Os autores mostraram que o algoritmo pode melhorar significativamente a precisão dos algoritmos para agregação de rótulos. Também trabalhamos com penalidades no MktPoints, nele o trabalhador pode perder pontos relacionados a sua reputação se realizar uma tarefa sem a devida qualidade.

3.3.2.4 Atenção aos prazos e redes sociais

Nos sistemas de *crowdsourcing*, a qualidade dos relatórios sensoriais pode ser afetada por atributos sociais subjacentes e pelo egoísmo dos indivíduos. Isso levou Ren et al. [152] a considerar os impactos sociais e a confiabilidade dos usuários no processo seletivo dos participantes que irão realizar as tarefas de sensoriamento móvel. Nesse trabalho, os pesquisadores propõe o SACRM para seleção dos participantes mais adequados e alocação das recompensas relativas às tarefas. Especificamente, o sistema considera os atributos sociais, os atrasos nas tarefas e a reputação. Uma avaliação dos relatórios e um esquema de gratificação também foram implementados para medir a qualidade dos relatórios e recompensar tarefas com base na qualidade avaliada. Além disso, desenvolveu-se um sistema de gestão da reputação para avaliar a relação de confiabilidade e a taxa de desempenho versos custo dos trabalhadores. Uma análise teórica e simulações extensivas demonstram que o SACRM pode melhorar de forma eficiente um utilitário *crowdsourcing* e efetivamente estimular os participantes a melhorarem a qualidade dos seus relatórios de sensoriamento. Devido ao foco em validação de sensoriamento, não consideramos a solução viável para tarefas que requeiram a computação humana.

O *social participatory sensing* (SPS) é um novo paradigma que tenta resolver as limitações do *participatory sensing*, aproveitando as redes sociais *online* como uma infraestrutura. Uma questão crítica para o sucesso deste paradigma é assegurar a confiabilidade das contribuições fornecidas pelos participantes. Amintoosi e Kanhere [15] propõem um *framework* de reputação com aplicação agnóstica para SPS. A estrutura considera tanto a qualidade da contribuição quanto o nível de confiabilidade do participante dentro da rede social. Logo, esses dois aspectos são combinados através de um sistema de interferência para chegar a uma classificação final de confiança para uma contribuição. A pontuação

da reputação também é calculada para cada participante como uma resultante dos *ratings* de confiança que lhe são atribuídos. O método adota o algoritmo PageRank como alicerce para o módulo de reputação. Simulações extensivas demonstram a eficácia da estrutura na atribuição das pontuações de confiança e reputação. O MktPoints tem a intenção atribuir pontuações a reputação, mas uma comparação com o trabalho de Amintoosi e Kanhere [15] seria falha por não termos a intenção de chegar a um valor exclusivo para esse fim. Porém, um estudo de correlação estatística pode ser alvo de trabalhos futuros.

3.3.2.5 Solução distribuída

Para Josang, Ismail e Boyd [98] ainda há mais um aspecto a ser considerado. Uma vez que existem ambientes em que um sistema de reputação distribuído, ou seja, sem nenhuma função centralizada, é mais adequado que um sistema centralizado. Em um sistema distribuído, não há local central para enviar classificações ou obter classificações de reputação de outros usuários. Em vez disso, existem provedores distribuídos onde as classificações são recebidas, ou cada participante simplesmente registra a opinião sobre cada experiência com outras partes e fornece essas informações a pedido de terceiros. Uma parte confiante, que pensa em negociar com uma determinada parte alvo, deve encontrar os provedores distribuídos ou tentar obter classificações do maior número possível de membros da comunidade que tenham experiência direta com essa parte alvo. Nesse tipo de solução, a terceira parte confiável calcula a pontuação da reputação com base nas classificações recebidas. No caso de a parte confiável ter experiência direta com a parte alvo, a experiência desse encontro pode ser levada em consideração como informação privada, possivelmente com um peso maior do que as classificações recebidas.

Os dois aspectos fundamentais dos sistemas de reputação distribuída, segundo Josang, Ismail e Boyd [98], são (1) um protocolo de comunicação distribuído que permite aos participantes obter classificações de outros membros da comunidade; e (2) um método de cálculo da reputação usado por cada agente individual para obter notas de reputação das partes alvo com base nas classificações recebidas e, possivelmente, considerando outras informações. Veremos mais adiante que o MktPoints se aproxima das definições apresentadas por Josang, Ismail e Boyd [98].

Capítulo 4

Arquitetura Snow Flurry

Neste capítulo apresentaremos a arquitetura Snow Flurry através de cada um dos elementos que a compõe, seu projeto visa a escalabilidade e o suporte à mobilidade para sistemas de *crowdsourcing*.

4.1 Elementos da arquitetura

A Figura 4.1 apresenta uma visão geral da arquitetura Snow Flurry. Nela, os elementos opcionais estão representados por bordas tracejadas que envolvem a rede de distribuição de conteúdo, o serviço de reputação e a comunicação direta entre dispositivos (ilustrada pelos dois pares de setas à esquerda). Logo, os componentes mandatórios são: o serviço de notificação, a retaguarda e o dispositivo móvel.

4.1.1 Serviço de notificação

Projetamos a Snow Flurry para funcionar com um serviço de notificação que utiliza o método *push*. Esses serviços permitem o envio de mensagens ao usuário, dispensando o mesmo de tomar a iniciativa de verificar a existência de novas mensagens. Além disso, já vimos que o método *push* aumenta o número de tarefas concluídas e reduz a sobrecarga, mas ele também serve como incentivo de acesso à plataforma segundo Neyman [135] e Mc Mahon [122] devido aos princípios da Caixa de Skinner que veremos mais adiante no Capítulo 6.

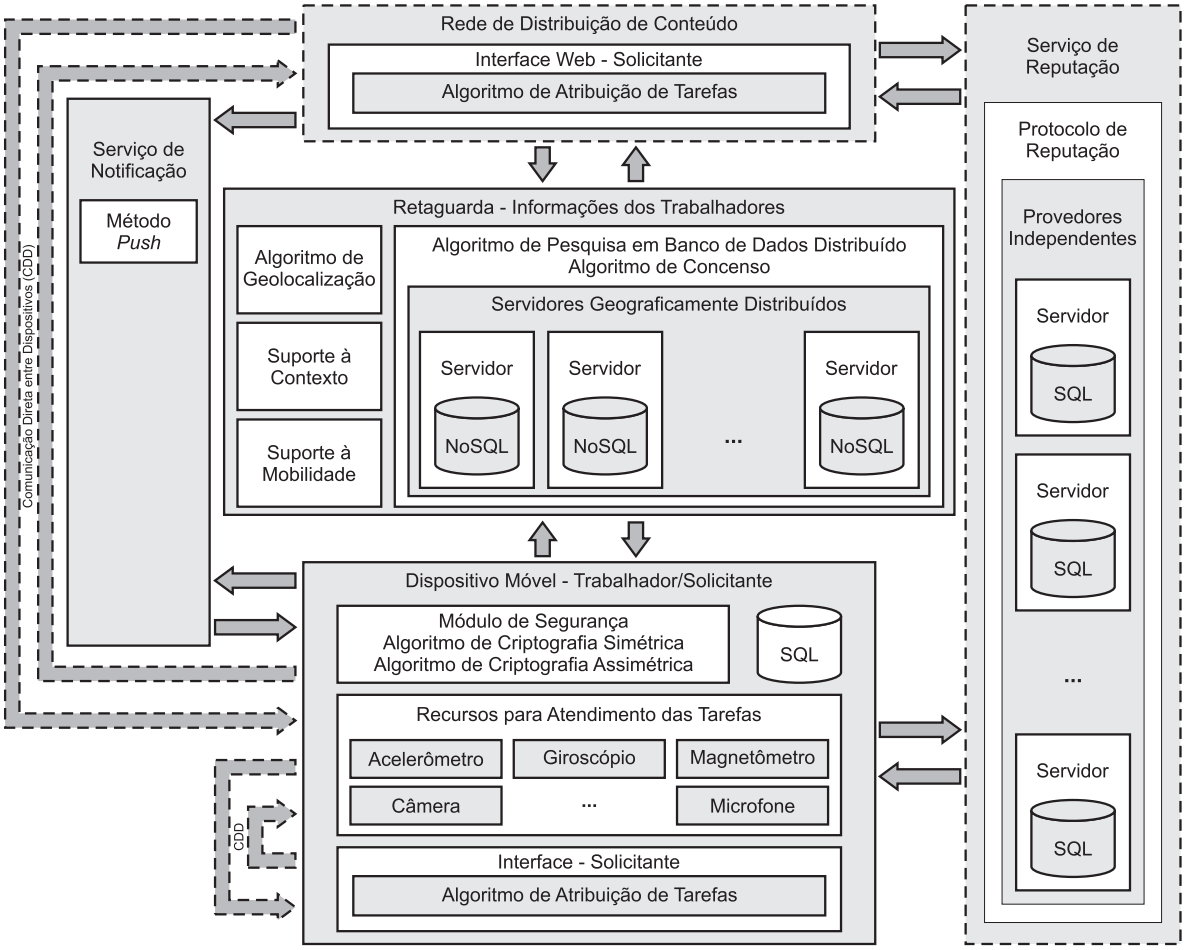


Figura 4.1: Arquitetura Snow Flurry

4.1.2 Retaguarda

A retaguarda é o componente responsável por manter as informações dos trabalhadores, do cadastro (com nome, *e-mail*, geolocalização e etc) até o resultado do trabalho realizado por eles em alguns cenários. Esse item se torna critico quando pensamos em um sistema de *crowdsourcing* de alcance global que deve suportar milhões de usuários espalhados pelo mundo. Utilizar uma arquitetura cliente/servidor provida por um supercomputador resultaria em uma solução com ponto crítico de falha e diversos níveis de qualidade de serviço, visto que a latência percebida pelos usuários iria variar conforme a localização geográfica de cada, ou seja, quanto mais afastado do servidor, pior seria o tempo de resposta da aplicação. Uma solução para esses pontos é distribuir servidores geograficamente.

4.1.2.1 Servidores geograficamente distribuídos

A distribuição geográfica dos servidores resolve duas questões: (1) evita que o sistema tenha ponto crítico de falha e (2) reduz a latência, melhorando a qualidade do serviço oferecido ao usuário. O primeiro item não trata de uma criação de réplicas simplesmente, isto é, não temos a intenção de espalhar supercomputadores pelo mundo, pois essa solução teria um alto custo. Logo, as informações também são distribuídas. Isso não significa que inexistiram réplicas de dados, porém a tomada de decisão de onde essas réplicas serão constituídas não é uma tarefa trivial. Precisamos de uma solução que mantenha a redução da latência ao posicionar a fonte de dados próxima de quem necessita dessas informações, provendo mais servidores (réplicas) quando a demanda por esses dados aumentar. Para resolver essas questão podemos utilizar algoritmos de consenso.

4.1.2.2 Algoritmo de consenso

No momento da realização de uma réplica e durante a manutenção dos dados replicados, precisaremos recorrer ao algoritmo de consenso. Basicamente, ele serve para manter os dados equalizados entre as réplicas, ou seja, sempre que uma alteração ocorre em um servidor é necessário gravar essa alteração nos outros que contem a réplica do dado alterado. Porém, podem ocorrer alterações simultâneas em servidores distintos e pra decidir qual alteração deve ser mantida é que utilizamos os algoritmos de consenso. Eles servem para obter um acordo entre processos sobre determinado dado, o que a maioria concordar como válido será mantido na base de dados. Podemos notar, nesse contexto,

que algoritmos para localização dos dados também são uma questão relevante em um sistema distribuído.

4.1.2.3 Algoritmo de pesquisa em banco de dados distribuído

Já vimos até aqui que os servidores que compõe o sistema não são réplicas exatas entre si, ou seja, os dados estão espalhados. Para localizar uma informação em um cenário assim, devemos fazer uso de algoritmos de pesquisa em banco de dados distribuído. Distribuir os dados de um banco entre diversos servidores trás benefícios e algumas consequências, essa operação pode requerer um grande esforço da equipe de desenvolvimento quando se utiliza bancos de dados relacionais. Esses bancos podem apresentar problemas devido a quebra da lógica de relacionamento que é um dos grandes benefícios dessa tecnologia. Nesses casos, os desenvolvedores precisaram resolver problemas relacionados a partição dos relacionamentos, como, por exemplo, realizar interações (*joins*) entre tabelas distribuídas. Uma alternativa é o uso dos bancos de dados orientados a documentos (NoSQL).

4.1.2.4 Servidores com NoSQL

Os bancos de dados orientados a documentos resolvem o problema das interações evitando as mesmas através da desnormalização, mesmo assim elas são possíveis pelo uso de chave-valor, família de colunas e grafos. Esses bancos também permitem uma estrutura mais dinâmica (como o JSon) que pode ser utilizada para definição de contextos, como dizer que um trabalhador ligado ao sistema de *crowdsourcing* é taxista, fotografo ou mesmo os dois. Isso nos leva a outra necessidade da arquitetura Snow Flurry, o suporte à contexto.

4.1.2.5 Suporte à contexto

Arquiteturas de uso geral que são suportadas por bancos de dados relacionais precisam de complexas soluções de parametrização para atender a necessidade de "adição de novos campos" quando a necessidade dos mesmo se revelar ou produzir uma nova versão do sistema a cada remodelagem do banco de dados. O suporte à contexto, em conjunto com os bancos de dados NoSQL, pode prover uma nova estrutura alinhada as necessidades do sistema sempre que mudanças se mostrarem necessárias.

4.1.2.6 Algoritmo de geolocalização

Os bancos de dados NoSQL mais modernos já oferecem pequisa por geolocalização, porém uma geolocalização eficiente em bases distribuídas ainda requer algum esforço por parte da equipe de desenvolvedores. Por conta disso, algoritmos de geolocalização especializados ainda precisam ser produzidos para cada cenário emergente.

4.1.2.7 Suporte à mobilidade

Aqui mobilidade é a capacidade do dispositivo trocar de identificador sem perder as conexões estabelecidas, ou seja, ser capaz de trocar de IP e, consequentemente, de rede sem ter que reiniciar uma operação (como, por exemplo, um *download* em andamento). Logo, entendemos que a retaguarda deve oferecer suporte à mobilidade para permitir que tanto o trabalhador quando o solicitante estejam em trânsito durante a realização das tarefas e transmissão das respostas em sistemas de *mobile crowdsourcing*.

4.1.3 Dispositivo móvel

Nos sistemas de *crowd sensing* os *smartphones* são mais requisitados devido a baixa interação com o usuário e os múltiplos recursos ofertados por esses aparelhos como GPS, acelerômetro e câmeras, por exemplo. Nos sistemas de *mobile crowdsourcing* podemos considerar outros tipos de dispositivos, como *tablets* e *notebooks*, por exemplo, pois o interesse maior está no poder cognitivo do usuário. Também entendemos que esses dispositivos devem permitir a solicitação de tarefas no sistema, ou seja, podem ser utilizados tanto por trabalhadores quanto por solicitantes.

Ainda que *notebooks* sejam um exemplo de dispositivo móvel, eles não se enquadram neste item da arquitetura por não suportarem, por padrão, a execução de *apps*.

4.1.3.1 Recursos para atendimento das tarefas

Como citado anteriormente, o ideal é que o dispositivo do trabalhador conte com GPS, acelerômetro, câmeras, entre outros recursos para atendimento das tarefas. O que direciona, mas não limita, ao uso dos *smartphones*. Do ponto de vista do solicitante, esses recursos são dispensáveis. Logo, ele pode utilizar um dispositivo mais simples.

4.1.3.2 *Interface* do solicitante no *app*

Além de viabilizar uma abertura de campanhas, a *interface* do solicitante no *app* também trás consigo uma estratégia que viabiliza a escalabilidade em sistemas de *mobile crowdsourcing*, a distribuição do algoritmo de atribuição de tarefas. Ao embarcar esse algoritmo no *app*, tiramos da retaguarda o ônus da centralização do processo de atribuição de tarefas e todo esforço computacional envolvido nesse evento, diluindo essa carga entre os dispositivos dos solicitantes.

4.1.3.3 Banco de dados local com SQL

O *app* precisa contar com um banco de dados relacional local para viabilizar o acompanhamento das tarefas, tanto do ponto de vista dos trabalhadores quanto para a gestão dos solicitantes. Essas informações só existiram nos dispositivos finais, ou seja, não serão mantidas na retaguarda do sistema.

4.1.3.4 Módulo de segurança

Um dos modelos de comunicação mais seguros que existe é o uso combinado de algoritmos de criptografia simétricos com assimétricos. O *app* deve viabilizar a criação de chaves criptográficas assimétricas e o cadastramento do usuário na retaguarda com o consequente envio da chave pública criada para a mesma. Posteriormente, atualizações na base devem ocorrer por meio de comandos assinados por algum algoritmo de criptografia simétrico, essa assinatura deve ser criptografada com a chave privada gerada pelo algoritmo criptográfico assimétrico antes do envio com o comando para a retaguarda.

4.1.4 Comunicação direta entre dispositivos

Na parte inferior e a esquerda da Figura 4.1 podemos ver um par de setas com bordas tracejadas que abraçam a sigla CDD que significa comunicação direta entre dispositivos. Ela ilustra um evento onde o dispositivo móvel do trabalhador se comunica diretamente com o dispositivo móvel do solicitante, por isso as setas saem e retornam ao componente dispositivo móvel.

Outra representação no lado esquerdo da Figura 4.1 são as setas com bordas tracejadas que conectam o dispositivo móvel ao componente rede de distribuição de conteúdo.

Elas ilustram uma comunicação direta entre um solicitante que está fazendo uso de uma *interface web* e um trabalhador que faz uso do *app*.

A CDD é um item opcional na arquitetura, ela pode ser utilizada para evitar custos relacionados ao armazenamento de dados na retaguarda.

4.1.5 Rede de distribuição de conteúdo

Outro item opcional na arquitetura Snow Flurry é o componente governado pela rede de distribuição de conteúdo. Essas redes permitem uma entrega com menor latência devido a distribuição geográfica de conteúdo estático, deixando o mesmo mais próximo de quem o solicita. O conteúdo estático em questão aqui é uma página para Internet que embarca o algoritmo de atribuição de tarefas. Essa página pode ser utilizada pelo solicitante para abertura de campanhas, rodando um processo local de atribuição de tarefas para desafogar a retaguarda (seguindo a estratégia já mencionada na Seção 4.1.3.2).

4.1.6 Serviço de reputação

O mecanismo de incentivo e/ou sistema de reputação são elementos opcionais na arquitetura Snow Flurry. Adicionar ou não essas funcionalidades à aplicação vai depender de cada tipo de necessidade. O desacoplamento dessas funcionalidades também favorece o uso de variadas soluções pela arquitetura Snow Flurry. Nesta tese, idealizamos uma arquitetura escalar para um sistema de reputação que está atrelado a um mecanismo de incentivo. Essa arquitetura deve se basear em provedores independentes suportados por um protocolo, onde as operações são registradas em bancos de dados do padrão SQL.

4.2 Comparativo entre arquiteturas do estado da arte

A seguir apresentamos a Tabela 4.1 que compara as arquiteturas do estado da arte com três variações do Snow Flurry. A Snow Flurry I é a arquitetura que foi implementada e que será apresentada no Capítulo 5. Ela utiliza nuvem para armazenar as respostas dos trabalhadores. A Snow Flurry II dispensa a nuvem e utiliza comunicação direta entre dispositivos, mas pra isso ela necessita de um elemento central chamado Espelho UDP que fornece um IP visível na Internet para que os dispositivos possam se comunicar. Veremos mais detalhes no Capítulo 5. A Snow Flurry III dispensa a nuvem e o elemento central por

utilizar um serviço global de nomes para resolver esses pontos. Ainda há outras variações possíveis, mas vamos nos limitar as essas três soluções.

Na Tabela 4.1 o traço "-" indica "não é o caso" ou "não foi informado", mesmo assim podemos ver que a Snow Flurry é melhor que as outras arquiteturas, exceto no item "Anonimato" que será alvo de trabalhos futuros.

Tabela 4.1: Comparativo entre arquiteturas

	Elemento Centralizado	Rede de Distribuição de Conteúdo	Algoritmo Distribuído de Atribuição de Tarefas	Notificação com Método *Push*	Computação em Nuvem e/ou Elástica	Servidores Geograficamente Distribuídos	Replicação Integral da Base de Dados	Busca em Banco de Dados Distribuído	Busca por Contexto	Geolocalização	Método de Distribuição da Informação	Estrutura Dinâmica (NoSQL)	Suporte à Mobilidade	Criptografia	Anonimato	Aplicação e/ou Código Fonte Disponível
To, Shahabi e Kazemi [184]	S	-	N	N	N	-	-	-	-	S	-	-	-	-	-	N
Geocrowd [106]	S	-	N	N	N	-	-	-	-	S	-	-	-	-	-	N
gMission [47]	S	-	N	N	N	-	-	-	-	S	-	-	-	-	-	N
CrowdBC [115]	N	-	N	-	N	S	S	S[1]	-	-	N	-	-	S	S	N
Réplicas [8]	-	-	N	-	S	-	S	N	-	S	N	-	-	-	-	N
mCrowd [198]	S	-	N	-	S	N	-	-	-	-	-	-	-	-	-	N
SANE [85]	-	-	-	-	-	S	S	N	-	-	-	-	-	-	-	N
DTN [7]	S	N	N	-	N	N	-	-	-	-	-	-	-	-	-	N
Borda [120]	N	N	-	-	S	S	S	N	-	S	N	-	-	-	-	N
Medusa [145]	S	N	N	-	S	-	S	-	-	S	N	-	N	-	-	S
Snow Flurry I	N	S	S	S	S	S	N	S	S	S	S	S	S	S	N	S
Snow Flurry II	S	S	S	S	N	S	N	S	S	S	S	S	S	S	N	S
Snow Flurry III	N	S	S	S	N	S	N	S	S	S	S	S	S	S	N	S

Capítulo 5

Implementação da Snow Flurry I

Neste capítulo apresentaremos a implementação e explicaremos o funcionamento da arquitetura Snow Flurry I, mostrando o papel de cada componente ao longo do processo e discutindo pontos em comum aos sistemas de MCS. Ao final discutiremos as versões II, III e algumas alternativas.

5.1 Componentes

A implementação da Snow Flurry I é composta por 4 componentes principais: um aplicativo para Android, o GNS Auspice, o Firebase Cloud Messaging e um meio de entrega suportado por dois serviços do Firebase, o Realtime Database e o Cloud Storage. Opcionalmente, a arquitetura pode contar com uma página para Internet, composta por HTML/JavaScript e distribuída por um serviço global de CDN, e um SR.

5.2 O processo

O círculo com o número dois na Figura 5.1 ilustra a busca por trabalhadores por parte do solicitante. Nesse momento, envia-se a geolocalização do centro do mapa em exibição que é utilizada como suposto ponto de interesse (PoI, do termo em inglês *Point of Interest*). Junto como o PoI é enviado um raio de 1 km (por padrão). O PoI muda conforme deslocamos o mapa. Ao receber os dados, a aplicação os apresenta no mapa, veja um exemplo na Figura 5.2, e refaz esse processo de busca por trabalhadores a cada 5 segundos (por padrão).

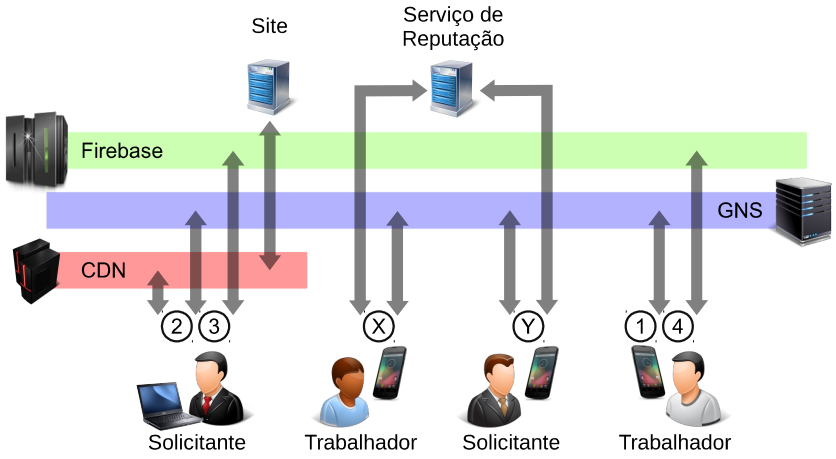


Figura 5.1: Implementação da arquitetura Snow Flurry I e indicação das etapas dos processos, da atribuição de tarefa ao recebimento dos resultados

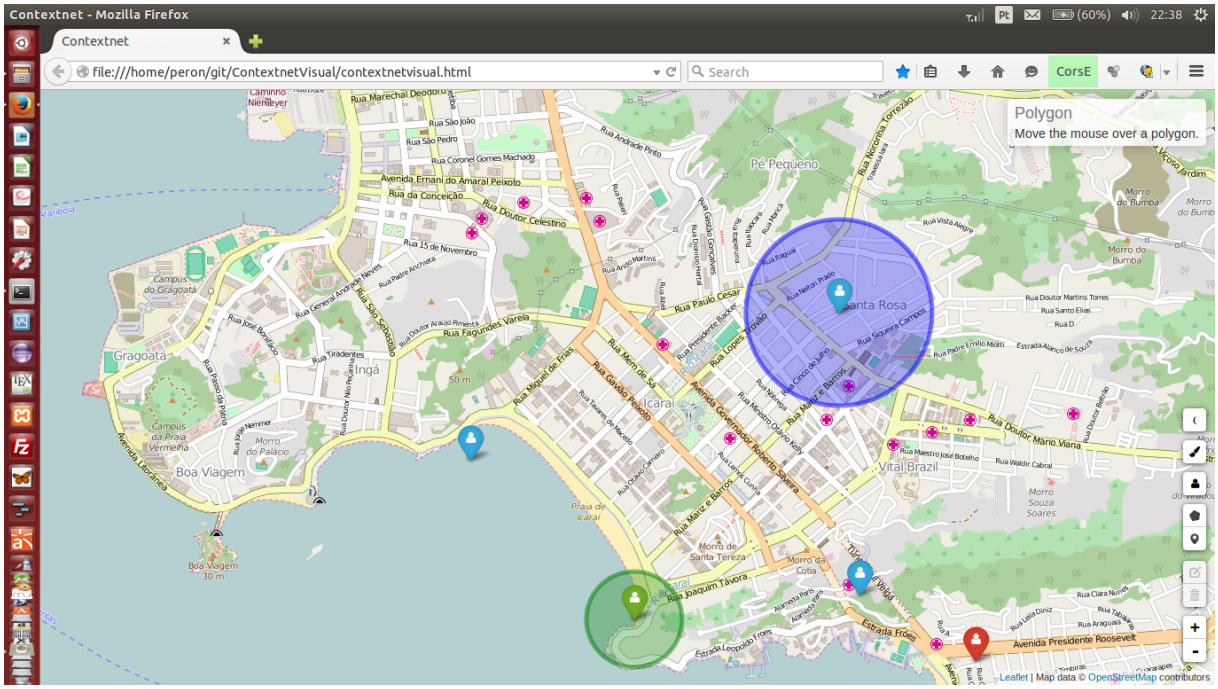


Figura 5.2: Visualizando a área de atuação do trabalhador

A Figura 5.2 também ilustra o desenvolvimento da página para Internet que utilizou o Leaflet.draw [99], uma biblioteca JavaScript de código aberto. Ela permite o desenho de formas vetoriais sobre o OpenStreetMap [76], um projeto de mapeamento colaborativo destinado a criar um mapa livre e editável do mundo. Na tela vemos cinco trabalhadores, três com o contexto azul, um verde e outro vermelho. Cada qual define o próprio raio de atuação e informa isso ao Auspice. O solicitante visualiza o perímetro derivado clicando sobre o marcador. No exemplo da Figura 5.2, se comparado ao trabalhador verde, o colaborar azul (localizado em Santa Rosa) atende uma área maior.

Continuando com o processo, escolhido o local, cria-se uma tarefa com determinado contexto e prazo para conclusão conforme ilustra o diagrama de MoLIC na Figura 5.3 e a tela na Figura 5.4. Nessa tela indicamos o contexto (cor) e adicionamos as informações necessárias à tarefa. Em seguida, clicamos em "Create" e depois no mapa para definir o local de interesse.

Também é possível notificar apenas os trabalhadores que estão dentro de determinada área, conforme diagrama de MoLIC da Figura 5.5. Então, o solicitante pode definir a tarefa para depois desenhar a área de interesse. Veja um exemplo na Figura 5.6, nela o trabalhador verde é notificado. Nesse caso, o contexto escolhido foi "Green". Por isso, os outros não foram acionados.

A notificação ocorre pelo Firebase Cloud Message, círculo número três da Figura 5.1. Em seguida, ocorre o evento indicado pelo círculo número quatro da Figura 5.1. Nele, o Firebase entrega a notificação ao dispositivo. Essa entrega usa o *token* que o aparelho recebeu do Firebase. Esse *token* é persistido no GNS pelo aplicativo para Android e coletado pelo solicitante, círculo um e dois da Figura 5.1. O GNS pode guardar *tokens* de outros serviços de notificação, como o Apple Push Notification Service (APNS) e o Microsoft Push Notification Service (MPNS). Optamos pelo Firebase, devido à integração com o ambiente de desenvolvimento Android Studio, mas existem outras formas do trabalhador ter acesso as tarefas. Algumas soluções utilizam o método *pull*, nele o trabalhador procura pela existência delas. O serviço de notificação do Firebase viabiliza o *push*, nesse método a tarefa é enviada ao trabalhador. Como citamos anteriormente, o *push* aumenta o número de tarefas concluídas e reduz a sobrecarga [102].

As tarefas são encaminhadas, preferencialmente, pelos serviços de notificação. Nesses serviços, estando o trabalhador desconectado, as tarefas/notificações são armazenadas por um período. Após essa espera, sem sucesso na entrega, a notificação é descartada. Caso o trabalhador tenha informado o *token* que o identifica nesses serviços para o Auspice, a aplicação enviará as notificações através deles. Dessa forma, caso o usuário deixe seu dispositivo desconectado por um longo período (férias, por exemplo), ele não receberá notificações antigas que tenham perdido o prazo estabelecido pelo solicitante.

Também podemos notificar os trabalhadores pelo aplicativo para Android. Na Figura 5.7, vemos como a solução se apresenta no aplicativo durante a etapa de pesquisa por trabalhadores (Figura 5.7(a)) e visão do raio de atuação dos mesmos, como ocorre na página para Internet (Figura 5.2). Da mesma forma, semelhante ao procedimento

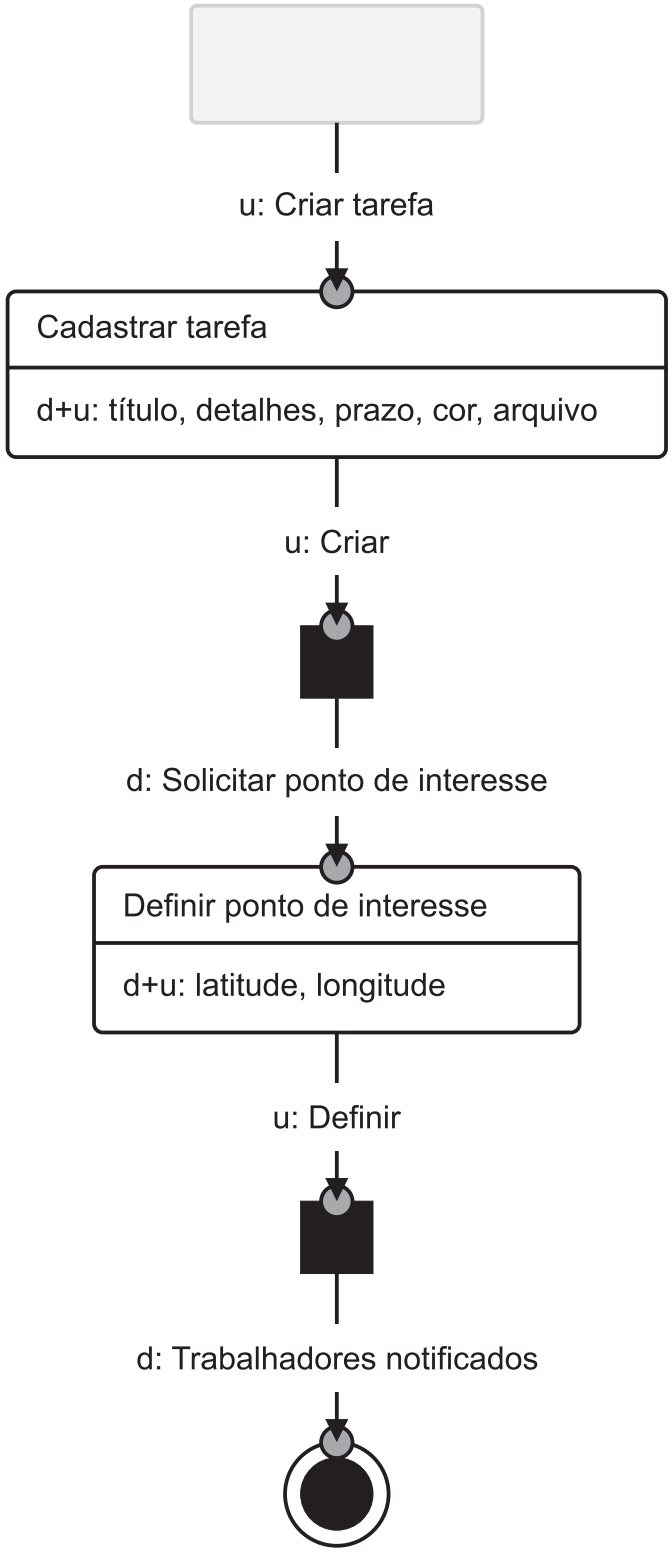


Figura 5.3: Diagrama da criação de tarefa por ponto de interesse

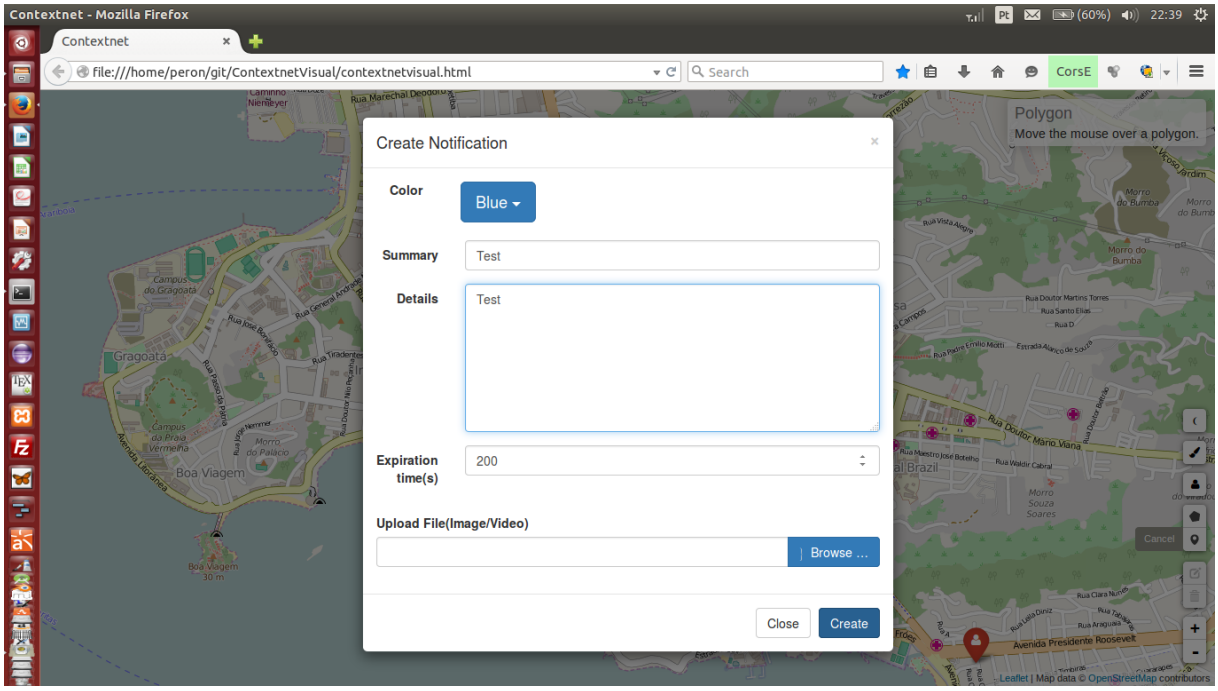


Figura 5.4: Caixa de diálogo que ilustra a criação de uma tarefa com definição de contexto (neste exemplo, a cor verde)

apresentado na Figura 5.4, podemos visualizar a abertura de uma tarefa por meio do aplicativo na Figura 5.7(b).

Na sequência, a Figura 5.8 mostra a visão do trabalhador no momento do recebimento da tarefa (Figura 5.8(a)) e na produção da resposta (Figura 5.8(b)). Por fim, a Figura 5.9 apresenta a página que o solicitante utiliza para visualizar as respostas na Internet.

O círculo número quatro da Figura 5.1 também representa a conclusão da tarefa, nesse momento o trabalhador envia os dados para o Firebase Realtime Database e/ou arquivos para o Firebase Cloud Store conforme ilustra o diagrama de MoLIC da Figura 5.10. Após o fim do prazo estabelecido para a tarefa, o solicitante recolhe as respostas, conforme passo que também é ilustrado pelo círculo número três da Figura 5.1. Dessa forma, ao utilizar a dupla Auspice/Firebase conseguimos: (1) focar no desenvolvimento das funcionalidades fins do MCS; (2) poupar linhas de código; (3) reduzir as configurações de ambiente; e (4) não precisamos monitorar e manter réplicas. Outra possibilidade é dispensar a persistência na nuvem, com isso, (5) não precisamos monitorar e manter instâncias. Esse último ponto será discutido no projeto da Snow Flurry II e III.

Do total de 15.842 linhas de código escritas para o desenvolvimento do núcleo das soluções apresentadas nesta tese, 2.744 foram destinadas a lógica do *site* e quase 11 mil para o aplicativo para Android (vide a Tabela 5.1). O aplicativo só atingiu uma quantidade

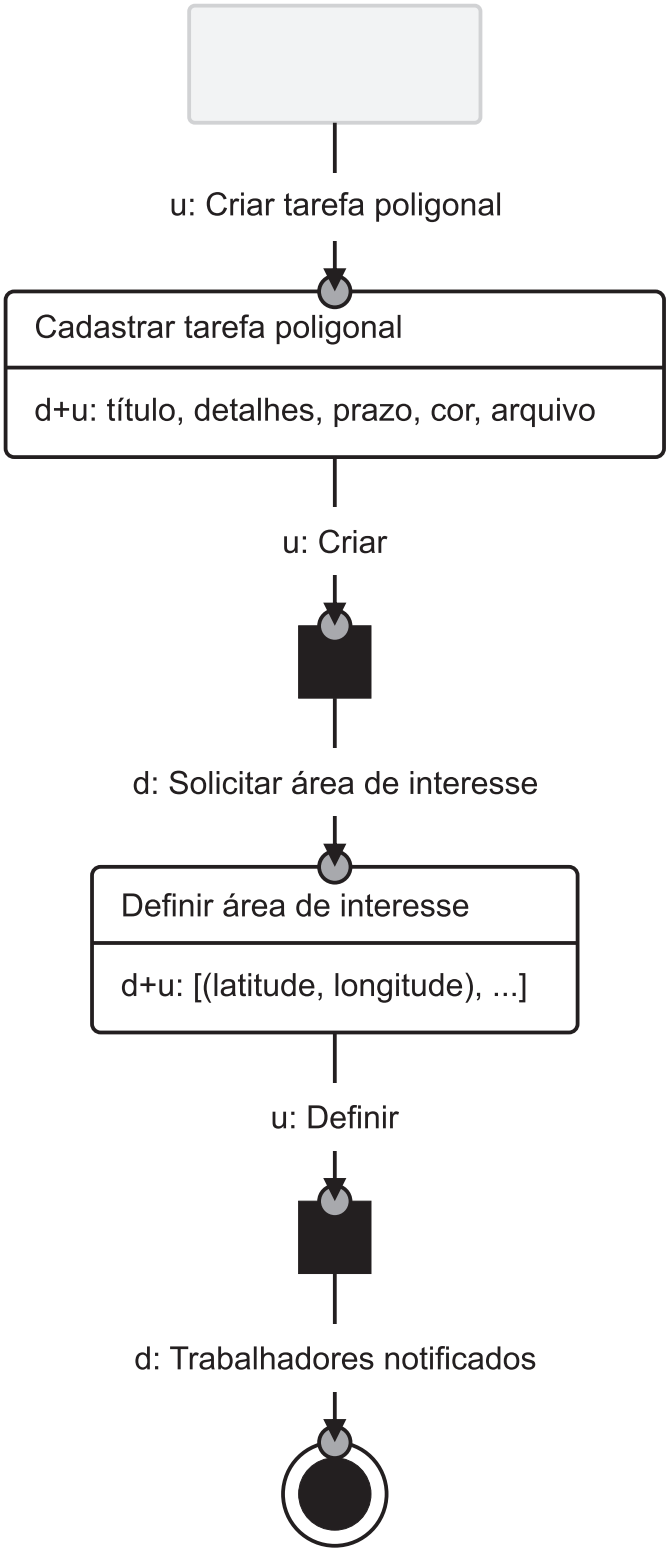


Figura 5.5: Diagrama da criação de tarefa por área de interesse

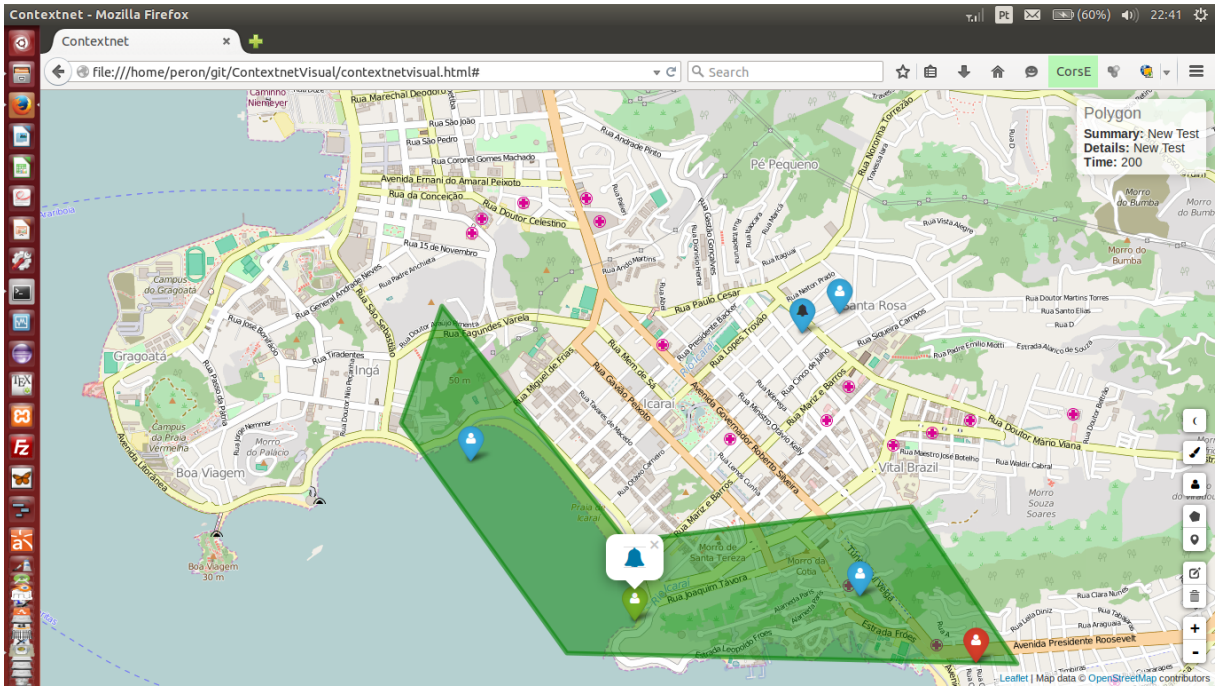


Figura 5.6: Notificando sem usar a área de atuação dos trabalhadores

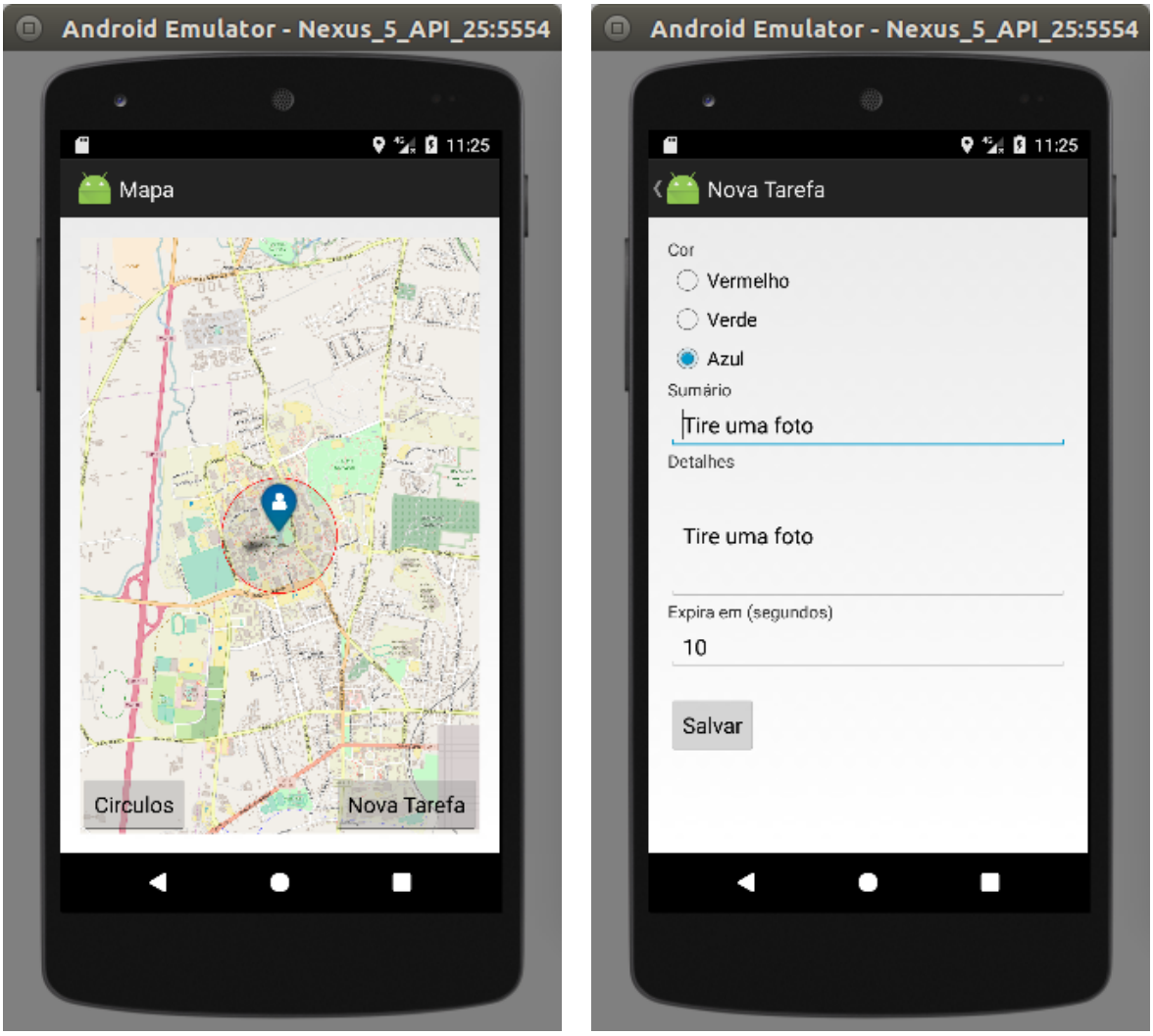
tão alta devido a impossibilidade de utilizar a biblioteca do MSocket em Android. Outro detalhe interessante que ocorreu durante a implementação foi a redução de 2/3, nas linhas de código, ao utilizarmos NodeJS no serviço de reputação.

Tabela 5.1: Linhas de código utilizadas na implementação das arquiteturas Snow Flurry I e II, com MktPoints

Soluções	**JavaScript**	**Java**
Site	2.744	0
Serviço de Reputação com Java	0	1.545
Serviço de Reputação com NodeJS	458	0
Serviço de Espelho UDP	101	0
App para Android	0	10.994
Total	3.303	12.539

5.3 Segurança

Segundo Kamienski et al. [101], muitos usuários não participam dos sistemas de *crowd sensing* e/ou *crowdsourcing* por questões de segurança. Divulgar a geolocalização é o maior dilema deles. Eles não desejam que seus dados fiquem públicos na rede, pois existe o receio de terceiros utilizarem os mesmos indevidamente. Uma medida de contorno é ocultar

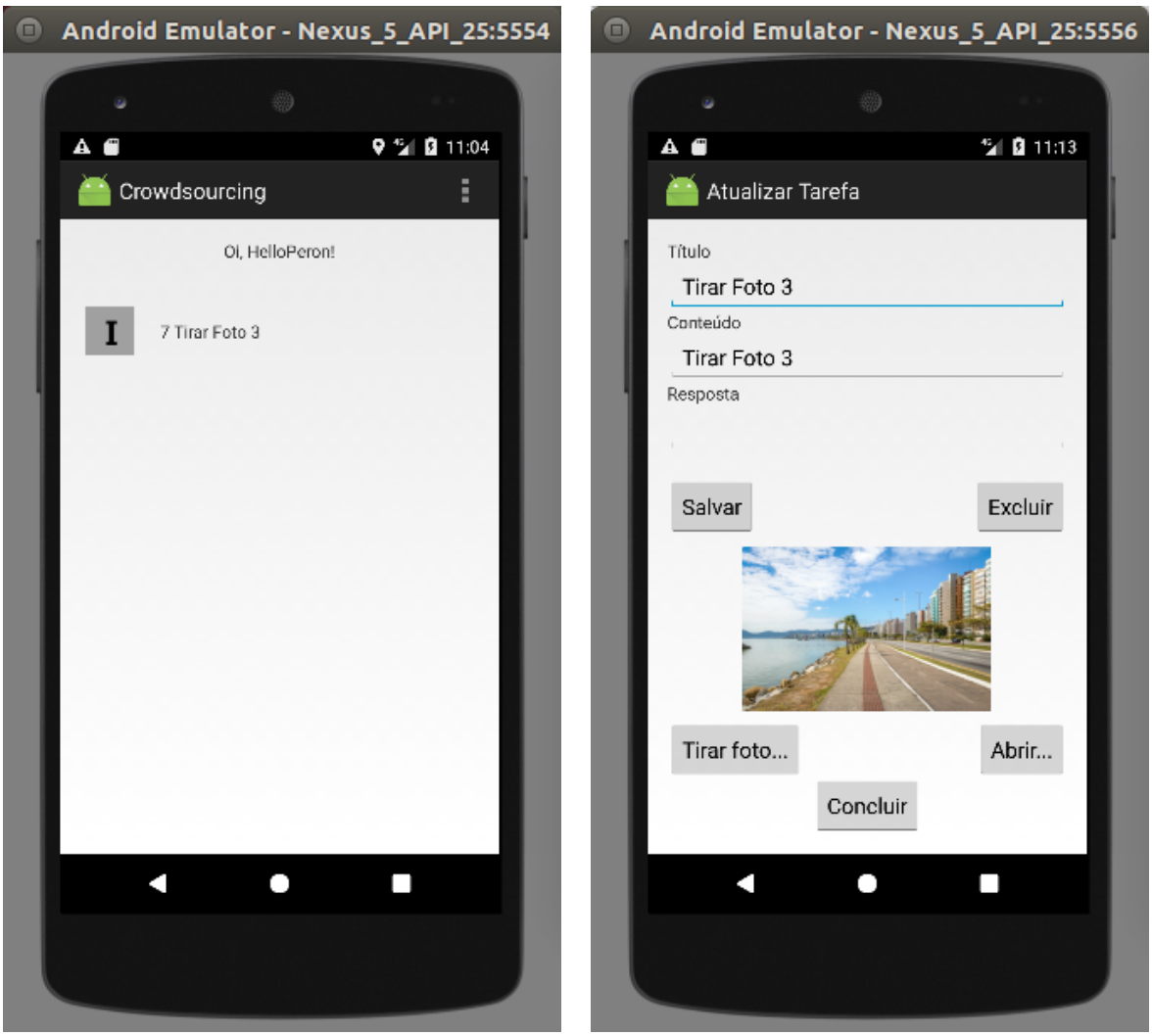


(a) Visualização de trabalhador no mapa com círculo indicando o raio de atuação

(b) Criando uma tarefa no aplicativo

Figura 5.7: Visão do solicitante no aplicativo

a identidade dos trabalhadores. Dessa forma, o provedor de serviço somente sabe que alguns participantes estão na região X no momento Y, mas não quem, exatamente [200]. A estratégia do Auspice é utilizar listas de controle de acesso (ACL, do termo em inglês *Access Control List*), de forma semelhante ao que é feito no Firebase, onde o usuário pode definir quem poderá ter acesso aos seus dados (com listas brancas, do inglês *whitelist*) ou quem não terá acesso (listas negras, do inglês *blacklist*). Apresentar maiores detalhes, sobre o funcionamento dessa e de outras técnicas, foge ao escopo deste trabalho.



(a) Chegada da tarefa (b) Respondendo a tarefa com foto

Figura 5.8: Visão do trabalhador no aplicativo

5.4 Prova de conceito

Para a prova de conceito desenvolvemos os dois tipos de recursos que consomem os serviços do GNS, a página para Internet e o aplicativo para dispositivos Android. Os solicitantes podem utilizar tanto a página para Internet quanto o aplicativo, já os trabalhadores apenas o aplicativo. A aplicação Android permite o cadastro no Auspice e a atualização dos dados, entre esses dados destacamos a geolocalização, as informações de contexto e o *token* do serviço de notificação. Essa comunicação é ilustrada pelo círculo com número um na Figura 4.1. Diferente de Peng et al. [137], que atualiza a retaguarda a cada 2 minutos, a Snow Flurry só encaminha uma atualização ao GNS se ocorrer uma mudança.

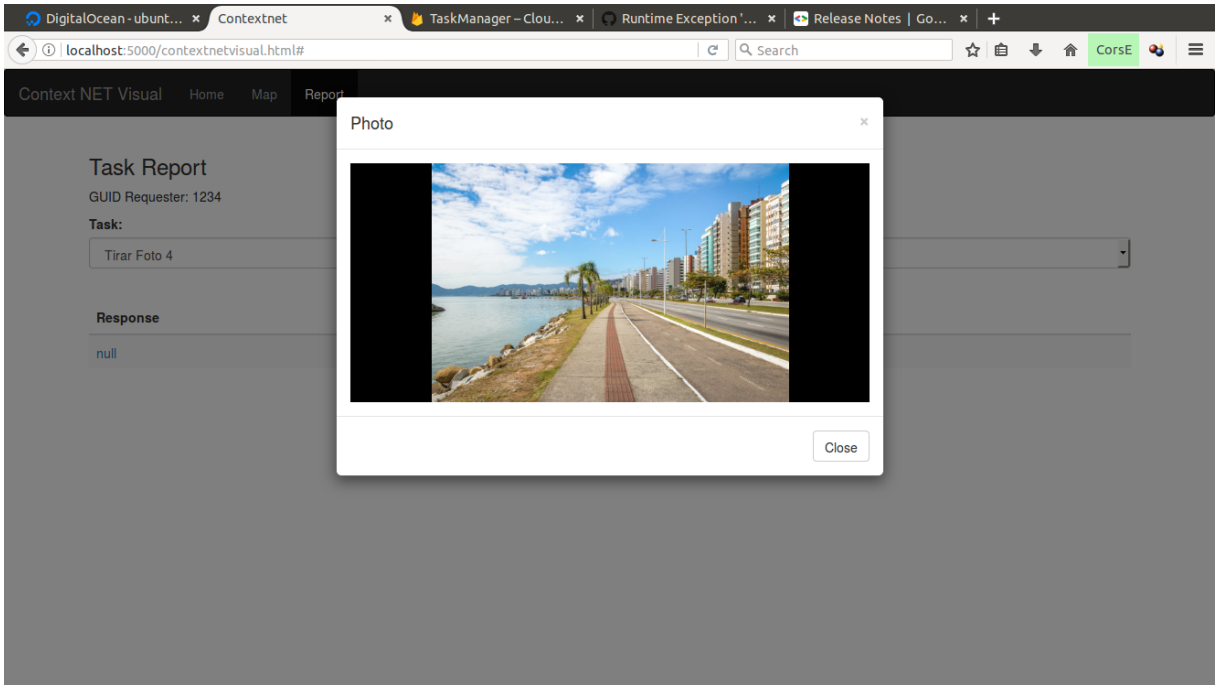


Figura 5.9: Visualização da resposta, com foto, na página *web*

5.5 Snow Flurry II e o serviço Espelho UDP

Implementamos um serviço simples, chamado Espelho UDP, para viabilizar comunicação direta entre dispositivos na arquitetura Snow Flurry II. Ele atua em conjunto com o GNS, criando um canal temporário em redes 3G, 4G e/ou Wi-Fi. O processo começa com uma modificação no evento número 3 da Figura 5.11. Na notificação, o solicitante deve incluir um horário estimado para início da coleta. O agendamento evita que vários trabalhadores enviem dados ao solicitante simultaneamente, saturando a banda do mesmo. Próximo ao horário agendado, o trabalhador contata o serviço de Espelho UDP. Evento representado pelo círculo A da Figura 5.11. Esse serviço devolve a dupla IP/Porta que é visível na Internet. Em seguida, ele inclui/atualiza essa informação no GNS (círculo B da Figura 5.11). No horário marcado o solicitante obtém do GNS as informações de IP/Porta dos trabalhadores, círculo C da Figura 5.11. Na sequência, ele envia uma mensagem de coleta a um deles e uma de aguarde aos demais, se for o caso. Repetindo essa última mensagem a cada 10 segundos (isso é necessário para manter a conexão aberta), círculo D da Figura 5.11. Por fim, o trabalhador seguinte é selecionado e o processo se repete até o fim da coleta.

A Snow Flurry II pode ficar ainda mais simples, dispensando o uso do Espelho UDP, ao utilizar um recurso do MobilityFirst conhecido como MSocket.

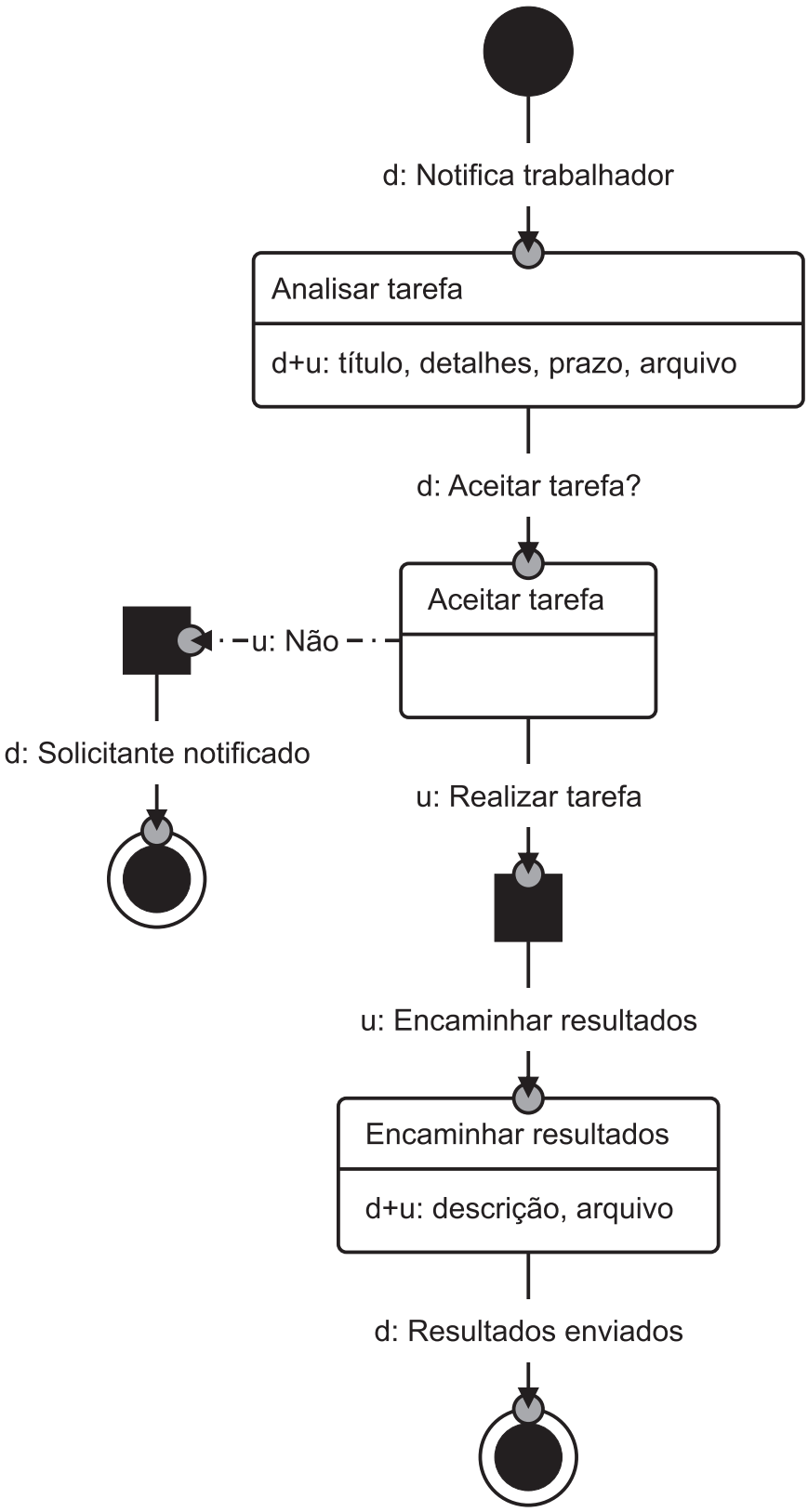


Figura 5.10: Diagrama ilustrando o processo de recebimento da tarefa e envio dos resultados por parte do trabalhador

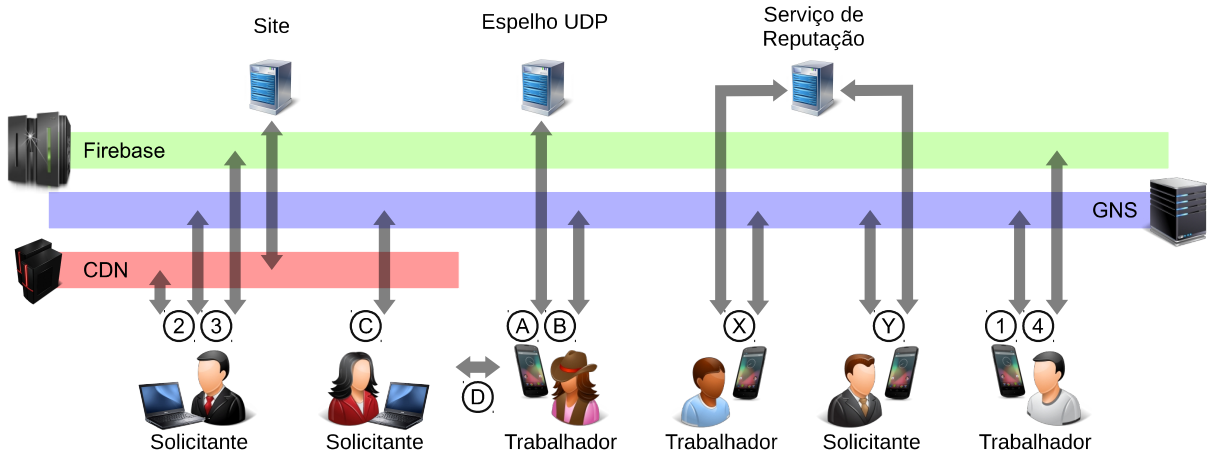


Figura 5.11: Implementação da arquitetura Snow Flurry II e indicação das etapas dos processos, incluindo a comunicação direta entre dispositivos

5.5.1 MSocket

Na comunicação direta, devido à mobilidade dos dispositivos envolvidos, existe a possibilidade de que uma das pontas ou ambas mudem de rede, trocando de IP. Para evitar a perda da comunicação, podemos usar o MSocket. Ele é uma biblioteca Java (análoga ao Socket convencional) que permite a mobilidade de ambas as extremidades sem perda de conexão. Esta solução é parte da pesquisa sobre a arquitetura MobilityFirst e trabalha em conjunto com o Auspice, usando chaves criptográficas na comunicação. Fora algumas adaptações adicionais no código, podemos simplesmente trocar Socket e ServerSocket, por MSocket e MServerSocket, respectivamente. Dessa forma, é possível prover mobilidade a uma aplicação que trabalhe com uma comunicação sobre TCP e isso sem depender de um DNS [31, 80]. Não utilizamos essa biblioteca em na implementação da Snow Flurry III, porque ela ainda precisa de algumas adaptações para rodar perfeitamente na plataforma Android. O suporte à mobilidade foi provido por uma API HTTP do Auspice.

5.5.2 Qual o custo da comunicação direta para os usuários?

Do ponto de vista do trabalhador, é necessário manter o dispositivo conectado, após a execução da tarefa, para transmitir os dados no horário programado. Isso pode ser percebido como um custo ou não, já que o trabalhador pode manter o dispositivo ligado na expectativa de receber novas tarefas. Para o solicitante, a recepção de dados é lenta se comparada ao modelo com retaguarda. Nas soluções de *crowd sensing* ainda podemos adicionar ao contexto a possibilidade de manipular e resumir os dados na retaguarda,

reduzindo ainda mais os custos assumidos pelo dispositivo do solicitante ou do trabalhador, dependendo de onde o processamento é realizado.

5.5.3 Alternativas: Mobile P2P e nuvem pública

Em arquiteturas *crowd sensing* é normal a busca pela automação total de todas as etapas, envolvendo minimamente o usuário. No *crowdsourcing* a participação do usuário é esperada de alguma forma. Com isso em mente, podemos considerar as possibilidades à seguir.

A comunicação direta com cada trabalhador da campanha pode tornar a transmissão de dados demorada do ponto de vista do solicitante e do trabalhador por "aguardar" numa fila. Para que o trabalhador não espere e o solicitante possa baixar o conteúdo mais rapidamente, podemos usar mais alguns trabalhadores para criar uma rede P2P semelhante ao BitTorrent, usando os GUIDs do GNS no lugar dos IPs para fornecer suporte à mobilidade. Nesse caso, ao concluir uma tarefa com uma carga de dados significativa, o trabalhador cria um arquivo .torrent e o envia ao solicitante e a um grupo de trabalhadores que ajudará na distribuição, convocados por meio de uma tarefa inversa. Dessa maneira, o trabalhador já inicia a transmissão imediatamente e o solicitante começa a baixar as partes do conteúdo de acordo com sua capacidade.

Outra possibilidade (intuitivamente mais econômica) é o trabalhador enviar o conteúdo para uma nuvem pública como o DropBox[1] onde é possível criar um link para o conteúdo e envia-lo ao solicitante ou utilizar o SendSpace[2] que dispensa cadastro e criação de link, mas requer *e-mail*.

5.6 Snow Flurry III e o Auspice

Por fim, existe a arquitetura Snow Flurry III que conta com a possibilidade de utilizar o próprio Auspice para armazenar registros e arquivos, dispensando assim o Realtime Database e o Cloud Storage utilizados na arquitetura Snow Flurry I. Em sua estrutura, o Auspice utiliza um banco de dados não relacional orientado a documento (o MongoDB). Devido a forma como ele foi implementado, podemos utilizar a mesma função utilizada para criar um contexto para armazenar dados em formato JSon. Com os arquivos sendo convertidos para Base 64, previamente. Dessa forma, a Snow Flurry III dispensa o serviço

[1] https://www.dropbox.com/
[2] https://www.sendspace.com/

de Espelho UDP da Snow Flurry II e não apresenta o problema da fila por persistir os dados no Auspice a qualquer tempo.

Capítulo 6

MktPoints: A implementação do mecanismo de incentivo e do serviço de reputação

Neste capítulo, apresentamos um novo mecanismo de incentivo que vai além da motivação extrínseca dos sistemas de leilão. O mecanismo MktPoints trabalha atrelado a um sistema de reputação e se apoia sobre quatro pilares: (1) qualificação das respostas; (2) gamificação; (3) marketing de rede; e (4) caixa de Skinner. A seguir veremos cada um deles.

6.1 Qualificação das respostas

Uma das operações no site Booking.com[1] é intermediar reservas entre a rede hoteleira e seus hospedes. Após uma estadia, quando a reserva é feita pelo *website*, o hospede recebe um *e-mail* que solicita a avaliação do hotel. Essa avaliação utiliza o MOS, nesse método a qualificação é representada por um número inteiro entre 1 (ruim) e 5 (excelente), conforme Tabela 6.1 [153]. No caso do Booking.com, a média das notas gera um *ranking* entre os estabelecimentos e ajuda no processo de escolha dos interessados em hospedagem. O MktPoints também utiliza MOS, a qualificação é feita com o envio da nota ao SR do trabalhador. Na arquitetura Snow Flurry podem existir diversos serviços de reputação, cada qual com um grupo de trabalhadores. Essa descentralização diminui o risco desse serviço se tornar um gargalo.

[1] https://www.booking.com/

Tabela 6.1: *Mean Opinion Score - MOS*

MOS	Qualidade
5	Excelente
4	Bom
3	Razoável
2	Pobre
1	Ruim

6.2 Gamificação

Segundo Garcin, Faltings e Jurca [78], uma tarefa fundamental em sistemas de reputação é agregar várias classificações de *feedback* em um único valor que pode ser usado para comparar a reputação de diferentes entidades. O *feedback* é mais comumente agregado usando a média aritmética. No entanto, a média é bastante suscetível a desvios e vieses e, portanto, pode não ser o agregado mais adequado. Diante disso, adotamos uma forma nova de utilizar o MOS e representar seu valor.

A gamificação ocorre quando atribuímos características de jogos a um sistema não-jogo. Segundo Silva et al. [169], isso promove envolvimento e motivação. O MktPoints não atribui a nota do MOS diretamente ao trabalhador, mas converte a mesma em pontos. Cada avaliação coloca em jogo 64 pontos e o trabalhador pode apostar mais, se desejar. Por exemplo, considere que um usuário conseguiu acumular 5 mil pontos. Ele pode gravar no GNS a dupla 2000/$1, isso significa que ele aposta 2K da reputação em troca de uma tarefa com valor mínimo de $1 (evento ilustrado pelo círculo X na Figura 4.1). Em outra mão, o solicitante pode definir uma dupla 1000/$2 que significa selecionar trabalhadores com no mínimo mil pontos de reputação, pagando no máximo $2. Ao selecionar um que atenda a essas características, o solicitante envia um pedido de reserva ao SR do mesmo (evento ilustrado pelo círculo Y na Figura 4.1). Existindo saldo, o SR debita o valor e confirma a reserva (note que isso limita o número de operações com apostas). Em seguida, o solicitante envia a tarefa ao trabalhador (esse pode rejeitar a tarefa para devolver, rapidamente, a reserva ao saldo). Depois de concluída a tarefa, o solicitante atribui um MOS e envia ao SR do trabalhador (também ilustrado pela comunicação do círculo Y na Figura 4.1). Por fim, o serviço converte o MOS seguindo a Tabela 6.2.

Em resumo, o trabalhador perde todo o valor apostado se receber uma nota 1. Recebe de volta a metade se ganhar um 2. Apenas cancela a reserva se levar um 3. Ganha 50% frente uma nota 4 e dobra o valor apostado com 5.

Tabela 6.2: Exemplo de pontuação com MOS

MOS	Retorno da Reserva	Pontos Ganhos
1	0	0
2	1000	0
3	2000	0
4	2000	1000
5	2000	2000

6.3 Marketing de rede

A seleção por pontos pode desmotivar o ingresso de novos trabalhadores, frente as altas apostas feitas pelos antigos. Para evitar que os novos comecem o jogo com zero pontos, o MktPoints utiliza a estratégia do convite. Com a ajuda dessa estratégia, *websites* como o Orkut[2] e o Facebook[3] obtiveram um rápido crescimento no número de usuários. Por exemplo, considere um veterano chamado John que possui 5.000 pontos e que convida Kevin para participar do MCS. Então, John informa o SR sobre o convite. Em seguida, Kevin se cadastra no SR que John utiliza. O SR informa Kevin que existe um convite de John. Na sequência, Kevin confirma o convite. Por fim, o SR credita na conta de Kevin a mesma quantidade de pontos que John possui. Isso resolve o problema de Kevin. Ele não vai iniciar no MCS com zero pontos. Também podemos entender que, nesse processo, John atesta a reputação de Kevin.

No estado da arte dos mecanismos de incentivo para MCS encontramos os leilões reversos que são fundamentados em motivação extrínseca (recompensa financeira). No exemplo apresentado aqui, John pode ver Kevin como um concorrente no mercado de MCS e uma ameaça a recompensa financeira. Logo, diferente do que ocorreu nas redes sociais, John não teria motivos para convidar Kevin ou outras pessoas. Porém, o modelo de negócio conhecido como Marketing de Rede resolve essa questão. Normalmente, nesse modelo cada participante recruta concorrentes e ganha com a venda desses e das pessoas que eles recrutam [54]. Trata-se de uma atividade ilegal semelhante aos Esquemas de Pirâmides [185]. O MktPoints utiliza esse modelo na propagação dos pontos, não nas recompensas financeiras. Por exemplo, considere que Kevin realizou uma tarefa que lhe rendeu 640 pontos. John vai receber 320 pontos por ter convidado Kevin. Quem convidou John receberá 160 pontos e assim sucessivamente até o sexto nível. Limitamos em seis

[2]O site foi desativado em 30 de setembro de 2014
[3]https://www.facebook.com/

níveis por questões de processamento, mas também porque, segundo a Teoria do Mundo Pequeno das Redes Complexas, em média, estamos distantes de qualquer outra pessoa no mundo por seis saltos [127].

Como não vamos aplicar comissão sobre o valor das tarefas, não utilizaremos percentuais. Isso evita que os pontos da reputação sejam fracionados. Vamos usar valores inteiros e limitar a distribuição até o sexto nível (veja a Tabela 6.3), mas a profundidade e a lateralidade da rede não serão limitadas. Adotaremos 64 pontos como a menor pontuação que uma tarefa pode ter. Tarefas maiores serão compostas por múltiplos não fracionados dessa pontuação mínima. Essas definições visam a redução do esforço computacional no processo de pontuação.

Tabela 6.3: Exemplo de distribuição dos pontos através dos níveis

Nível	Pontos
-	64
1	32
2	16
3	8
4	4
5	2
6	1

6.4 Caixa de Skinner

Em 1948, o psicólogo Burrhus Frederic Skinner criou um artefato para estudar a Lei do Efeito de Thorndike. Tratava-se de uma câmara de condicionamento operante que mais tarde ficou conhecida como "Caixa de Skinner". Ela continha um acionador que permitia, ao animal que estivesse confinado na mesma, sua utilização para obter algum tipo de recompensa. Sua estrutura variava um pouco conforme os objetivos, isso permitiu a observação do comportamento frente a variados graus de interação e recompensa. Tal recompensa provocava a liberação de dopamina no cérebro de quem a recebia. A dopamina é um neurotransmissor que está associado a sensação de prazer e sua liberação serve para (1) ensinar ao indivíduo que o resultado foi correto e (2) incentiva-lo a obter o mesmo resultado no futuro [125].

O princípio acima é muito utilizado em cassinos, games e redes sociais, onde a recompensa financeira não é uma certeza ou nem é o caso [65, 108]. Entendemos que um

mecanismo de incentivo deve levar isso em consideração, pois existem tarefas em sistemas MCS onde a recompensa financeira não está envolvida e, nesses casos, as soluções com leilão reverso do estado da arte perdem seu impacto.

O MktPoints utiliza o sistema de pontuação e o marketing de rede para alcançar o efeito de razão variável da Caixa de Skinner, pois os pontos podem se acumular livremente, tal qual as visualizações e curtidas no YouTube, e crescer inesperadamente, visto que o usuário pode receber pontos derivados de tarefas realizadas por pessoas da rede que ele formou ou pelo *feedback* de uma tarefa realizada por ele.

6.5 Considerações sobre *truthfulness*

Podemos encontrar na fronteira entre os temas "segurança" e "mecanismo de incentivo", os métodos de apoio à verdade. No MktPoints, a avaliação do MOS pode ajudar a inibir contribuições falsas devido ao risco de baixa avaliação (que gera perda de pontos). Devemos também ter em mente que a ambiguidade da tarefa ou sua natureza complexa pode resultar em contribuições que são difíceis de serem identificadas como falsas, mesmo se analisadas por um especialista. Outro ponto é que estamos lidando com tarefas que as máquinas têm dificuldade ou são incapazes de resolver. Portanto, seria um erro pensar em um método automatizado para verificar a integridade das respostas. Existem soluções que utilizam outra tarefa para confirmar uma primeira, mas corremos o risco de apenas validar o "lugar comum" e descartar as ideias daqueles que pensaram "fora da caixa".

6.6 Ataque Sybil

O trabalho de Hoffman, Zage e Nita-Rotaru [88] apresenta várias classes de ataques, as quais listamos a seguir:

- **Promoção de si mesmo** - Os atacantes manipulam sua própria reputação aumentando-a falsamente.
- **Lavagem branca** - Os atacantes escapam da consequência de abusar do sistema usando alguma vulnerabilidade do sistema para reparar sua reputação. Depois de restaurar sua reputação, os invasores podem continuar com o comportamento malicioso.

- **Calúnia** - Atacantes manipulam a reputação de outros nós, informando dados falsos para diminuir a reputação dos nós da vítima.
- **Orquestração** - Os atacantes orquestram seus esforços e empregam várias das estratégias acima.
- **Negação de serviço** - Os atacantes causam negação de serviço, impedindo o cálculo e a divulgação de valores de reputação.

Como meio para atingir os objetivos da maioria das classes relatadas por Hoffman, Zage e Nita-Rotaru [88] existe uma técnica que é conhecida como ataque Sybil. Nesse tipo de ação, um invasor subverte o sistema de reputação criando um grande número de identidades falsas e as usa para obter algum tipo de benefício no sistema [134]. Nos trabalhos de Drucker e Fleischer [67] e Emek et al. [69] encontramos análises sobre modelos similares ao MktPoints que utilizaram o marketing multinível em conjunto com as redes sociais. Os modelos trabalham com um tipo de tarefa viral, onde o solicitante incentiva os trabalhadores a promover um produto entre seus amigos, a fim de aumentar a receita de vendas. Enquanto Emek et al. [69] mostram que o mecanismo de recompensa geométrico satisfaz unicamente muitas propriedades desejáveis, exceto ser a prova de Sybil, Drucker e Fleischer [67] apresentam um mecanismo de recompensa nivelada que é localmente à prova de Sybil e à prova de colusão. O conluio aqui considera apenas a criação de nós falsos de forma colaborativa. Em todos os mecanismos de marketing multinível, a receita é gerada endogenamente pelos nós participantes e uma fração da receita é redistribuída sobre os referenciadores. Em tarefas ligeiramente diferentes, Conitzer et al. [51] propõem mecanismos que são robustos à manipulação de nomes falsos para aplicativos como o Facebook, convidando seus usuários a votar em seus termos de uso futuros. Além disso, podemos ver nos trabalhos de Yu et al. [202, 203] uma proposta de protocolo que limita a influência corruptora de ataques Sybil em redes P2P, explorando *insights* de redes sociais.

No entanto, Nath et al. [134] destacam que em redes de *crowdsourcing* reais, haveria um custo diferente de zero envolvido na criação de nós falsos e, portanto, deveria haver um ponto de inflexão para que o ganho líquido do trabalhador aumentasse até ele criar muitos nós Sybil, mas começaria a diminuir depois disso. Note que é impossível para um trabalhador computar o ponto de inflexão *a priori*, pois sua própria recompensa é incerta no momento em que ele é recém-recrutado por alguém e ele tenta criar nós Sybil. Portanto, diante dessa incerteza, o trabalhador pode se assegurar de um arrependimento limitado se decidir não criar nenhum nó Sybil.

O MktPoints poderia utilizar o envio de *token* por SMS[4] para desencorajar a criação de nós Sybil, mas ainda existiria a possibilidade do conluio que, num primeiro momento, poderia partir da própria rede criada em um movimento onde os membros se avaliariam mutuamente para aumento dos próprios pontos. A identificação desse tipo de comportamento e outras questões relacionadas ao ataque Sybil serão verificadas em trabalhos futuros.

6.7 Comparativo entre mecanismos de incentivo do estado da arte

Segundo Zhang et al. [207], os incentivos são divididos em três categorias: entretenimento, serviço e dinheiro. Os métodos baseados em entretenimento tentam motivar os participantes normais, transformando certas tarefas de detecção em jogos, de modo que os participantes possam experimentar enquanto estão contribuindo para os sistemas de detecção. Integrar tarefas de detecção e jogos depende muito da estrutura dos jogos, o que resulta em uma aplicabilidade limitada. Incentivos de serviço estão na forma de troca de contribuição pessoal por serviços do sistema. Normalmente, esse tipo de incentivo pode ser amplamente aplicado nos sistemas de serviço público, como monitoramento da poluição do ar, monitoramento de ruído e monitoramento de tráfego. Mecanismos de incentivo monetário proporcionam aos participantes os incentivos mais intuitivos. Quando os participantes contribuem para as tarefas de detecção, eles podem receber algum dinheiro como recompensa. Neste paradigma, grandes quantidades de tempo livre dos participantes podem ser utilizadas e as tarefas de detecção em si não precisam possuir propriedades como diversão ou conforto de serviço. Em outras palavras, os mecanismos monetários são mais gerais do que os mecanismos de entretenimento e serviço, uma vez que as duas últimas classes de mecanismos devem ser implementadas de uma maneira específica no aplicativo.

Considerando o exposto por Zhang et al. [207], apresentaremos nas Tabelas 6.5 e 6.6 um comparativo entre mecanismos de incentivo que podem ser enquadrados na categoria dinheiro. Na Tabela 6.4 encontramos as legendas necessárias para o entendimento das siglas que aparecem nas Tabelas 6.5 e 6.6. O que podemos notar com o comparativo é que apenas seis mecanismos atendem a todos os critérios apontados e, entre eles, somente o MktPoints não favorece uma das partes em detrimento da outra e tem foco na atração e retenção dos trabalhadores.

[4]https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html

Tabela 6.4: Legendas da Tabela 6.5

Sigla	Descrição
Mecanismo	
EP	Envio de preço
LOS	Leilão de oferta de selo
LR	Leilão reverso
LMA	Leilão multiatributivo
JS	Jogo de Stackelberg
LRO	Leilão reverso online
Esquema de Pagamento	
U	Uniforme
V	Variável
F	Fixo, baseado em ranking
MLP	Menor lance perdido
L	Lances
LP	Limite de pagamento
PI	Parte igual da recompensa total
PP	Parte proporcional da recompensa total

6.8 Comparativo entre serviços de reputação do estado da arte

Os serviços de reputação podem ser classificados pelas fontes de informação que eles usam para inferir confiança ou reputação como segue [142, 158, 160, 159]:

- As experiências diretas são uma das fontes mais valiosas de informação para os agentes. O autor diferencia entre interações diretas (ID) e observações diretas (OD).
- Informações de testemunhas (IT) são informações coletadas de outros agentes. Embora esse item em particular possa ser estendido com uma tipologia completa de informações de testemunhas, como observações de terceiros, interações com terceiros, comunicação de reputação etc., o trabalho permanece nesse nível.
- A informação sociológica (IS) é baseada na análise das relações sociais entre os agentes e pode ser computada através da análise de redes sociais se houver dados relacionais disponíveis.
- O preconceito (P) é uma fonte de informação que permite a inicialização da confiança e a reputação quando nenhuma outra informação está disponível, e que coincide com

a noção humana sem levar em consideração a conotação negativa. A estereotipagem é uma noção muito relacionada que também tem sido usada nesse sentido.

As siglas apresentadas (ID, OD, IT, IS e P) orientam o entendimento da Tabela 6.7, no quesito fontes de informação. Analisando a Tabela 6.7, podemos ver que o MktPoints é o único que atende a todos os critérios apontados.

6.9 Comparativo entre sistemas de recomendação do estado da arte

Segundo Tintarev e Masthoff [182], os sistemas de recomendação tem objetivos variados que podem ser sintetizados na Tabela 6.8. Na Tabela 6.9, podemos ver uma comparação entre diversos sistemas de recomendação e observar que o MktPoints é o sistema que apresenta o maior número de objetivos descritos na Tabela 6.8, apesar de não ser o objetivo desta tese apresentar um sistema de recomendação. Isso fica mais claro quando analisamos o único item que não é atendido pelo MktPoints, a persuasão, visto que não existe a intenção da solução em recomendar algum tipo de contratação aos solicitantes, por exemplo.

Tabela 6.5: Comparativo entre mecanismos de incentivo

Autor	Objetivo da Plataforma	Mecanismo	Esquema de Pagamento	Confiança	Racionalidade Individual	Maximizar a Utilidade
Musthag et al. [133]	verificar incentivo monetário	EP	U,V	-	-	-
Reddy et al. [150]	verificar incentivo monetário	EP	F	-	-	-
Danezis, Lewis e Anderson [58]	verificar incentivo monetário	LOS	MLP	✓	-	-
Lee e Hoh [112]	minimizar e estabilizar o custo da plataforma; impedir que os usuários abandonem as tarefas de detecção	LR	L	-	-	-
Jaimes, Vergara-Laurens e Labrador [92]	maximizar a atuação dentro do orçamento; considerar a atuação espacial de usuários selecionados	LR	L	-	-	-
Yang et al. [199]	lucratividade	LR	LP	✓	✓	✓
Subramanian, Kanth e Vaze [175]	lucratividade	LR	LP	✓	✓	✓
Krontiris e Albers [111]	avaliação multiatributo de usuários	LMA	L	-	-	✓
Duan et al. [68]	recrutar usuários suficientes; análise do equilíbrio de Nash	JS	PI	-	-	-
Yang et al. [199]	equilíbrio único de Nash	JS	PP	-	-	✓
Zhao, Li e Ma [210]	competitividade	LRO	LP	✓	✓	✓
Koutsopoulos [110]	minimizar o custo, dada a restrição de qualidade	LR	LP	✓	✓	-
Singla e Kause [171]	privacidade protegida, competitividade	LR	LP	✓	✓	✓
Zhang et al. [208]	competitividade	LRO	LP	✓	✓	✓
TBA [208]	lucratividade; competitividade	LRO	L	✓	-	✓
TOIM [208]	lucratividade; competitividade	LRO	L	✓	✓	-
TOIMAD [208]	lucratividade; competitividade	LRO	L	✓	✓	-

Tabela 6.6: Comparativo entre mecanismos de incentivo - continuação

Autor	**Objetivo da Plataforma**	**Mecanismo**	**Esquema de Pagamento**	**Confiança**	**Racionalidade Individual**	**Maximizar a Utilidade**
single-minded [96]	bem-estar social	LR	L	✓	✓	-
multi-minded [96]	bem-estar social	LR	L	-	✓	-
Koutsopoulos [110]	minimizar o custo	LRO	L	-	✓	-
Lee e Hoh [112]	minimizar o custo; bem-estar social	LR	L	-	-	-
OMZ [210]	minimizar o custo	LRO	L	✓	✓	-
OMG [210]	minimizar o custo	LRO	L	✓	✓	-
TRAC [75]	localização do usuário no momento da atribuição das tarefas	LR	L	✓	✓	-
no-regret [172]	minimizar o custo e o arrependimento	LRO	L	-	-	✓
BP-UCB [172]	minimizar o custo	LRO	L	-	-	✓
MAB-MDR [94]	minimizar o custo e o arrependimento	LRO	L	-	✓	-
Singer e Mittal [170]	minimizar o custo	LRO	LP	-	-	✓
Kamar e Horvitz [100]	-	LRO	L	✓	-	-
Yang et al. [199]	centrado na plataforma	JS	PI	-	-	✓
Yang et al. [199]	centrado no usuário	LR	L	-	✓	-
Zhang e Schaar [209]	bem-estar social; gamificação; eficiência de Pareto	-	-	-	-	-
Lv e Moscibroda [119]	atrair e reter trabalhadores	-	-	-	✓	-
MktPoints	atrair e reter trabalhadores; recompensa variável; gamificação	EP	U	✓	✓	✓

Tabela 6.7: Comparativo entre serviços de reputação

Modelo	Cognitivo	Fontes de Informação	Visão Global	Verdade	Descentralizado	Generalidade
Abdul-Rahman e Hailes [2]	-	ID, IT	-	-	✓	-
AFRAS [36]	-	ID+IT	-	-	✓	✓
BDI+Repage [141, 143]	✓	ID+IT	-	✓	✓	✓
Carter, Bitting e Ghorbani [38]	-	IT	✓	-	-	N/I
Castelfranch e Falcone [39, 73]	✓	-	-	✓	✓	✓
Dirichlet [98]	-	IT	✓	-	-	N/I
eBay[5]	-	IT	✓	-	-	N/I
Esfandiari e Chandrasekharan [71]	-	ID, OD, IT, P	-	✓	✓	✓
FIRE [90]	-	ID+IT	-	✓	✓	✓
Foner [142]	-	N/I	N/I	-	-	N/I
ForTrust [87]	✓	-	-	✓	✓	✓
Histos [206]	-	ID+IT	-	-	✓	N/I
LIAR [131]	-	ID+OD	-	✓	✓	-
Marsh [121]	-	ID	-	✓	✓	✓
Mui, Mohtashemi e Halberstadt [129]	-	ID, IT	-	-	✓	✓
Padovan et al. [136]	-	ID+IT	-	-	✓	N/I
Rasmusson e Jason [147, 146]	-	IT	✓	-	-	N/I
Regan e Cohen [151]	-	ID+IT	-	✓	✓	-
Regret [157]	-	ID+IT+IS+P	-	✓	✓	✓
Repage [156]	✓	ID+IT	-	-	✓	✓
Ripperger [154]	-	-	-	✓	✓	-
Schillo, Funk e Rovatsos [161, 162]	-	ID, OD, IT	-	✓	-	✓
Sen e Sajja [164]	-	ID, OD, IT	-	-	✓	-
Sierra e Debenham [168]	-	ID+IT+IS	-	✓	✓	✓
Sporas [206]	-	IT	✓	-	-	N/I
Yu e Singh [201]	-	ID, IT	-	✓	✓	-
SWORD [205]	-	-	-	✓	-	N/I
Allahbakhsh et al. [12]	-	ID + P	-	-	-	N/I
Yu et al. [204]	-	-	-	-	-	N/I
Jagabathula, Subramanian e Venkataraman [91]	-	ID	-	✓	-	N/I
SACRM [152]	-	ID + IS	-	-	-	N/I
Amintoosi e Kanhere [15]	-	ID + IS	-	-	-	N/I
Josang, Ismail e Boyd [98]	-	ID	✓	-	✓	N/I
MktPoints	✓	ID, OD, P	✓	✓	✓	✓

Tabela 6.8: Legendas da Tabela 6.9

Objetivo	Definição
Transparência	Explicar como o sistema funciona.
Examinabilidade	Permitir que os usuários digam ao sistema o que está errado.
Verdade	Aumentar a confiança dos usuários no sistema.
Eficácia	Ajudar os usuários a tomar boas decisões.
Persuasão	Convencer os usuários a experimentar ou comprar.
Eficiência	Ajudar os usuários a tomar decisões mais rapidamente.
Satisfação	Aumentar a usabilidade ou a a diversão.

Tabela 6.9: Comparativo entre sistemas de recomendação

Sistema	**Transparência**	**Examinabilidade**	**Verdade**	**Eficácia**	**Persuasão**	**Eficiência**	**Satisfação**
Adrissono et al. [20]	-	-	✓	✓	-	-	-
Bilgic e Mooney [28]	-	-	-	✓	-	-	-
Billsus e Pazzani [29]	-	✓	-	✓	-	-	-
Swartout [176]	✓	-	-	✓	-	-	-
Cosley et al. [52]	-	-	-	-	✓	✓	-
Czarkowski [57]	-	✓	-	-	-	✓	-
Herlocker, Konstan e Riedl [86]	-	-	-	✓	✓	-	✓
McCarthy [123]	-	-	-	✓	-	✓	-
McGinty e Smyth [124]	-	-	-	-	-	✓	-
McSherry [126]	✓	-	-	-	-	✓	-
Pu e Chen [144]	-	-	✓	-	-	-	-
Sinha e Swearingen [173]	-	-	✓	-	-	-	-
Thompson, Göker e Langley [180]	-	-	-	✓	-	✓	-
Wärnestál [194]	-	-	✓	-	✓	-	-
MktPoints	✓	✓	✓	✓	-	✓	✓

Capítulo 7

Avaliação

Neste capítulo apresentamos uma comparação com o estado da arte em termos de performance. Inicialmente, não era objetivo desta tese obter uma latência menor, mas alcançar escalabilidade com menor custo e menor complexidade técnica. No entanto, veremos que a latência encontra uma relação com esses fatores.

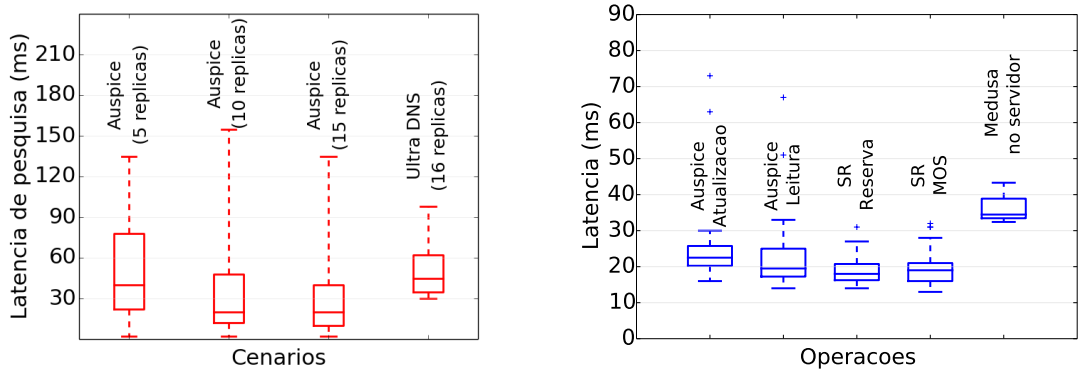
7.1 A prova de conceito e o experimento comparativo com o Medusa

Para validar a arquitetura Snow Flurry realizamos uma PoC, onde percorremos todo o processo construído para viabilizar o MCS. Primeiro, escolhemos o PoI e visualizamos os trabalhadores disponíveis em tempo real. Depois, elaboramos uma tarefa que consistia em fotografar e fazer uma breve descrição. Desde a solicitação à consulta das respostas, tudo transcorreu sem problemas, levando cada operação menos de 5 segundos. O cadastro e as atualizações das coordenadas do GPS junto ao GNS também ocorreram com sucesso, levando menos de 5 segundos. Nenhum dos elementos da arquitetura apresentou problemas de performance durante a PoC. O único ponto de atenção foi gerado pelo mecanismo de notificação do Firebase que, em algumas ocasiões, levou até 3 minutos para entregar a tarefa. Segundo o setor de suporte dessa aplicação, ela pode atrasar intencionalmente as mensagens para impedir que um *app* consuma recursos excessivos e prejudique a vida útil da bateria. O conflito de escolha entre consumo de bateria e entrega mais rápida de tarefas ainda requer estudos, incluindo se o tempo de 3 minutos é crítico para um aplicativo MCS. O site mystarbucksidea.com[1] mostrou que 95% de seus resultados são recebidos em

[1]https://ideas.starbucks.com/

até duas horas. Portanto, um tempo mínimo para notificação depende muito do tipo e da natureza da tarefa.

Nessa implementação da Snow Flurry para MCS, podemos concluir que a escalabilidade do sistema como um todo depende da escalabilidade de suas partes. Em função do que foi exposto até aqui, reconhecemos essa característica no Auspice e no Firebase. Sobre o Auspice, podemos ver uma comparação entre ele e um DNS na Figura 7.1(a), uma análise completa pode ser obtida no trabalho de Sharma et al. [167]. Já o Firebase é uma plataforma reconhecida no mercado. A implementação utilizou o Plano Spark (gratuito) do Firebase que limita o Realtime Database à 100 conexões simultâneas, 1 GB de armazenamento e 10 GB de tráfego por mês. Já o Storage tem limite de 5 GB de armazenamento e 1 GB de tráfego por dia, sendo também limitado a 20 mil operações de *upload* e 50 mil de *download* por dia. Exceder esses limites envolveria aumento de custos e de atividades relativas ao gerenciamento de instâncias. O Plano Spark não apresenta limites para o Cloud Messaging. Então, podemos manter o Firebase para fins de notificação e dispensar o Realtime Database e o Storage. Logo, sem esses elementos, podemos considerar a comunicação direta entre dispositivos para tornar o sistema escalável sem aumento de custos e/ou complexidade operacional. Porém, ainda falta avaliar a escalabilidade do novo mecanismo de incentivo e comparar o desempenho das comunicações de cada parte com outra solução do estado da arte.



(a) Auspice com 5 réplicas é comparável ao UltraDNS com 16 réplicas e com 15 réplicas obtém uma latência 60% menor [167].

(b) Comparação com o Medusa envolvendo o tempo de resposta sem o atraso da rede para as principais fases de comunicação da implementação da arquitetura Snow Flurry I com MktPoints.

Figura 7.1: Avaliação da latência dos componentes utilizados na implementação da arquitetura Snow Flurry I

7.1.1 Descrição do cenário

Para avaliar o SR, usamos duas máquinas Linode[2], cada uma composta por 1 núcleo de CPU com 64 bits, 2 GB de RAM, 30 GB de armazenamento SSD, 2 TB de transferência, 40 Gbps de entrada, 1000 Mbps de saída, executando o sistema operacional Ubuntu 16.04. Em uma das máquinas instalamos a versão 1.8.3-5 do XAMPP que operava o banco de dados no MySQL. Na outra, instalamos o Java versão 1.8.0_151 e o Node.js versão 8.11.1 que executou o SR. Num segundo momento, removemos o SR e instalamos o GNS em apenas uma delas. Em ambos os casos, havia uma carga de 5000 trabalhadores no sistema.

7.1.2 Análise dos resultados

Avaliamos o desempenho da implementação usando a latência como métrica e a aplicamos sobre cada tipo de comunicação. Para descartar atrasos relacionados à rede, as solicitações foram disparadas a partir da própria máquina onde a implementação se encontrava. Realizamos 30 repetições em cada caso e obtivemos os *boxplots* mostrados na Figura 7.1(b).

Podemos ver que os custos gerais são de apenas 34,47 ms em média para a Medusa, o que permite um atendimento de 1740 instâncias de tarefas por minuto com um único servidor. Os criadores do Medusa afirmam que esse componente é altamente paralelizável, pois cada instância de tarefa é relativamente independente e pode ter um "Controlador de Tarefa" separado. Assim, se a taxa de transferência de tarefas se tornar um problema, seria possível aproveitar os recursos elásticos da computação em nuvem para dimensionar o sistema [145]. Seguindo o mesmo princípio, podemos ver que a Snow Flurry é capaz de atender mais solicitações por máquina (vide Tabela 7.1).

Tabela 7.1: Avaliação da latência e ganhos derivados dos componentes utilizados na implementação da arquitetura Snow Flurry I

Requisições	**ms/req**	**%**	**req/min**	**%**
Atualização de dados no GNS	25,40	26,31	2362	35,75
Leitura de dados no GNS	23,43	32,03	2560	47,13
Reserva de pontos no SR	19,40	43,72	3092	77,70
Registro de MOS no SR	20,00	41,98	3000	72,41
Abertura de campanha no Medusa	34,47	-	1740	-

[2]https://www.linode.com/

Na segunda coluna da Tabela 7.1, podemos ver o tempo, em milissegundos, necessário para atendimento de cada tipo de requisição. Os tempos médios de 25,4, 23,43, 19,4 e 20 ms, possibilitam, respectivamente, o atendimento de 2362, 2560, 3092 e 3000 requisições por minuto (quarta coluna da mesma tabela). Esses valores representando um aumento de capacidade entre 35,74% e 77,7% (quinta coluna), com uma diminuição na latência entre 26,31% e 43,71% (terceira coluna). Se a arquitetura Snow Flurry trabalhasse com computação elástica, esses dados já indicariam uma possível economia, considerando apenas a redução no provisionamento de recursos.

7.1.2.1 Complexidade

Ao contrário do Medusa, a Snow Flurry não precisa alocar recursos para lidar com o aumento de usuários e tarefas. O Auspice assume esse trabalho e ainda nos permite entregar uma latência menor devido à distribuição geográfica dos servidores, permitindo que os usuários se conectem ao servidor mais próximo.

O Medusa também pode ser distribuído geograficamente, mas vamos considerar o seguinte cenário: Até 2017, a MTurk possuía 500.000 trabalhadores espalhados por 190 países. Para atender a uma demanda semelhante, um sistema de MCS poderia posicionar pelo menos um servidor em cada país (visto que alguns países, devido ao tamanho ou volume de usuários, podem exigir mais servidores). Com isso, a equipe técnica teria que lidar com toda complexidade inerente a essa infraestrutura. Nesse contexto, entendemos por complexidade o trabalho necessário para manter essas máquinas sincronizadas, atualizadas e monitoradas 24 horas por dia, 7 dias por semana. Além de testadas e protegidas contra vulnerabilidades de segurança, com backups regulares e planos de restauração e contingência, entre outras operações. Ao usar o GNS, a implementação da Snow Flurry não sofre aumento de complexidade.

7.1.2.2 Custo

Além da complexidade e dos custos com manutenção de uma infraestrutura global, existem custos diretos relacionados à alocação das máquinas. Para ilustrar, vamos considerar o plano de hospedagem com computação elástica de menor valor por hora da Amazon: Com o EC2 t2.nano, teremos um custo de US$ 0,0058 por hora e uma máquina com 1 CPU e 0,5 GB. Isso gera uma despesa anual de US$ 50,11 por máquina. Se alocarmos uma estrutura global com 200 máquinas (arredondaremos os 190 países mencionados acima para simplificar), teremos um custo de US$ 10.022,00 por ano.

A arquitetura Snow Flurry não tem esses custos, porque eles são assumidos pelo Auspice. Mesmo quando utilizamos uma CDN, a Snow Flurry não escala os custos. A CDN pode ser provida por um dos planos gratuitos oferecidos no mercado, devido à forma compacta da implementação (um exemplo desse tipo de plano é o oferecido pela Cloudflare[3]). Quanto ao SR, ele pode trabalhar agregado ao sistema de pagamento ou usar outra forma de monetização. Assim, poderíamos ter parceiros independentes operando cada SR, mas compartilhando um mesmo protocolo (semelhante ao que ocorre com os provedores de *e-mail*). Apesar disso, o SR ainda poderia adotar uma arquitetura centralizada devido baixa probabilidade de acessos simultâneos.

No estudo de Schwartz et al. [163], encontramos referências ao número de trabalhadores por campanha (42 em média), campanhas abertas por dia (58 em média) e ao total de solicitantes (MTurk registrou, em 11 de março de 2015, 1634 solicitantes ativos). Portanto, podemos inferir: (1) operações simultâneas envolvendo 1500 ou mais solicitantes seriam atípicas e (2) com a infraestrutura apresentada, um único SR já seria capaz de atender a um sistema do tamanho de MTurk, se considerarmos que, seguindo uma distribuição de Poisson, a probabilidade de 90 ou mais solicitantes abrirem uma campanha em qualquer dia é de 0,007%.

7.1.3 Recomendações à UMass

Ao longo desta tese identificamos erros e pontos de melhoria que foram relatados a equipe que desenvolve o Auspice. Atualizações do Auspice ocorreram ao longo desta tese. Isso nos levou a realização de alguns testes e um deles envolveu a atualização da geolocalização por até 5000 trabalhadores simultâneos. Chegamos à 3000 trabalhadores, mas o GNS travou na sequência e, após o reinicio, ele continuou apresentando lentidão mesmo sem tráfego concorrente. Com até 3000 trabalhadores o comportamento permaneceu estável, conforme pode ser visto na Figura 7.2. Os histogramas apresentaram apenas uma aproximação de 3.5 à 1.0, conforme a quantidade aumentava, Figura 7.3. Continuamos acompanhando as atualizações do Auspice até a versão 1.19.16[4]. Porém, a nova versão do GNS apresentou um problema de performance onde a pesquisa por geolocalização levava em média 5 segundos para responder quando operando sobre uma base com 5000 registros, sem trafego concorrente e sem atraso de rede (fazendo a requisição no próprio servidor).

[3]https://www.cloudflare.com/pt-br/plans/
[4]https://github.com/MobilityFirst/GNS/releases/download/v1.19.16/GNS-1.19.16.tgz

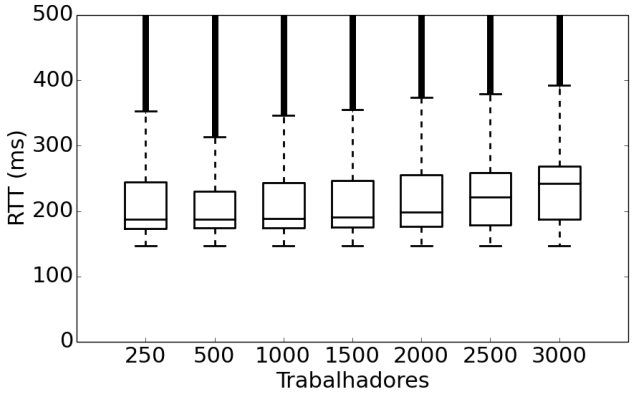
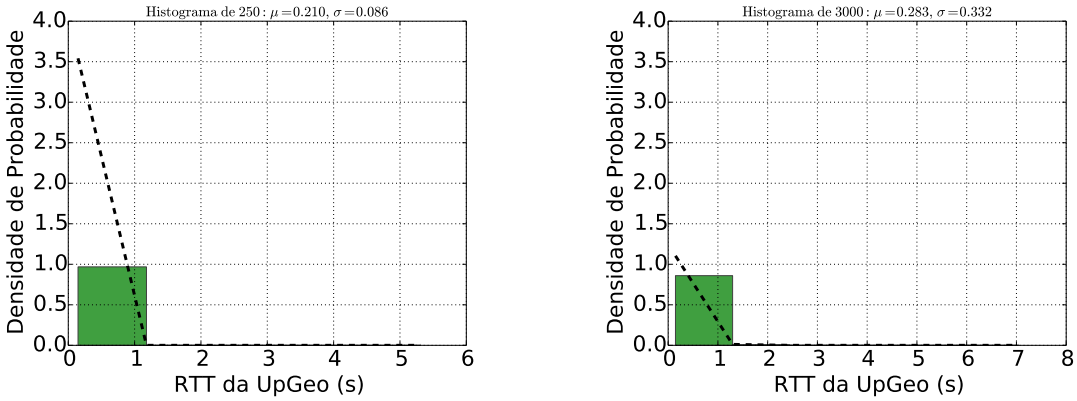


Figura 7.2: Avaliação do tempo de resposta do Auspice na etapa de geolocalização dos trabalhadores



(a) 250 trabalhadores

(b) 3000 trabalhadores

Figura 7.3: Deslocamento da média e aumento da variância no tempo de resposta do Auspice na etapa de geolocalização dos trabalhadores

7.2 Observações complementares em experimentos de escalabilidade com o MktPoints

Nesta seção vamos descrever o cenário utilizado para avaliar a capacidade do SR. Na sequência, vamos apresentar os resultados obtidos nos testes e discutir pontos da implementação que afetam a performance.

7.2.1 Descrição do cenário

Para avaliar o SR, utilizamos três máquinas virtuais em dois serviços diferentes de nuvem. Sendo duas máquinas da Linode[5], cada uma composta por 1 núcleo de CPU, 2 GB de

[5]https://www.linode.com/

RAM, 30 GB de armazenamento em SSD, 2 TB de transferência, 40 Gbps de entrada, 1000 Mbps de saída e ambas localizadas em Newark, NJ, USA. E outra da DigitalOcean[6], composta por 1 núcleo de CPU, 1 GB de RAM, 20 GB de armazenamento em SSD, 1 TB de transferência e localizada em San Francisco, CA, USA. Todas com sistema operacional Ubuntu 16.04 de 64 Bits. Em uma das máquinas da Linode instalamos o XAMPP versão 1.8.3-5 que operou o banco de dados em MySQL. Na outra, instalamos o Java versão 1.8.0_151 e o Node.js versão 8.11.1 que executaram o Serviço de Reputação. Na DigitalOcean, instalamos a mesma versão do Java e executamos um programa que emulava os solicitantes. Intencionalmente, a máquina que dispara as requisições (DigitalOcean em San Francisco) foi posicionada distante geograficamente das máquinas que sustentam o serviço (Linode em Newark).

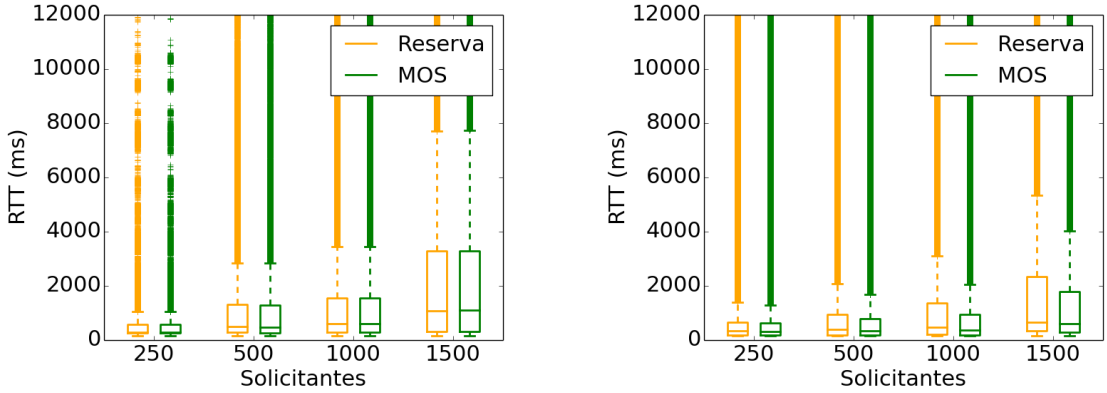
Cada solicitante iniciou uma campanha com 200 trabalhadores, selecionando os participantes aleatoriamente em um cadastro com 5000 registros. Depois, o solicitante realizou duas operações: (1) solicitou a reserva de pontos e (2) submeteu o MOS. Essas operações foram realizadas em sequência para cada trabalhador, um por vez. O experimento contou com grupos de 250, 500, 1000 e 1500 solicitantes operando simultaneamente. Cada rodada durou 10 minutos e foi repetida 30 vezes para cada grupo. Esses valores foram escolhidos com base na literatura. No estudo de Schwartz et al. [163] encontramos referências quanto ao número de trabalhadores por campanha (42 em média) e total de solicitantes (o MTurk registrou, em 11 de março de 2015, 1634 solicitantes ativos). Em outros trabalhos vemos referências ao número de trabalhadores (de 5 à 5000) [64, 163].

7.2.2 Análise dos resultados

O SR foi desenvolvido inicialmente em Java, utilizando uma comunicação simples por *socket* e uma solução *multithread* no atendimento das requisições. Porém, quando o número de conexões simultâneas aumentou o sistema começou a apresentar falhas. Neste trabalho, usaremos o termo "resposta negativa" para qualquer falha técnica no processamento da requisição. Durante o experimento, todas as respostas, sejam elas positivas ou negativas, foram recebidas em até 128 segundos. Distribuímos esses dados em 10 grupos com intervalos de 12800 milissegundos para analisar a frequência acumulada. Entre as respostas positivas, 90% ou mais se encontram na faixa de 0 à 12800 milissegundos. Isso foi observado em todos os quatro grupos (250 à 1500 solicitantes). As Figuras 7.4(a) e 7.4(b) foram construídas com esse subconjunto. Dividimos essa primeira faixa em outros 10 gru-

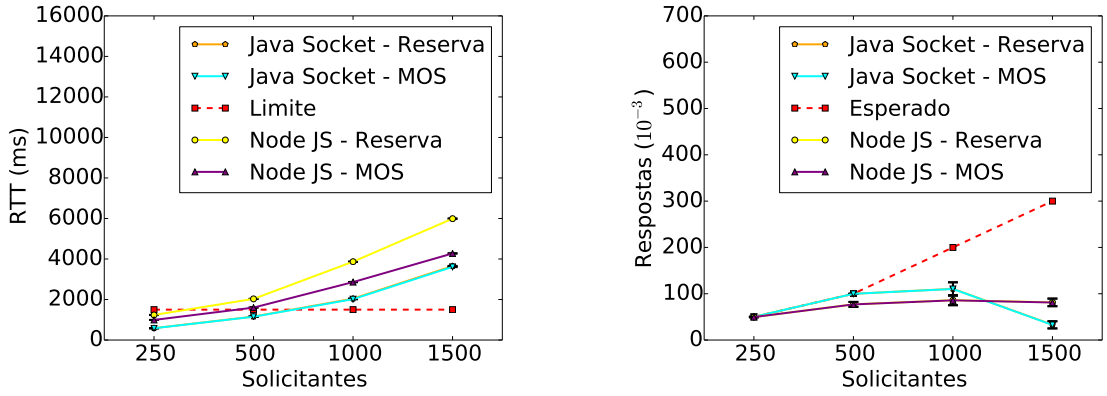
[6]https://www.digitalocean.com/

pos com intervalos de 1280 milissegundos. Essa nova separação indica que 90% ou mais dos eventos ocorreram em até 7680 milissegundos (nas seis primeiras faixas). Por esse motivo, e para melhor visualização, os histogramas apresentados nas Figuras 7.5, 7.6, 7.7 e 7.8 só mostram as respostas positivas que chegaram em até 7680 milissegundos. Esse teto, dos valores mais expressivos, também pode ser visualizado no diagrama de caixa das Figuras 7.4(a) e 7.4(b). Note o extremo superior, ele termina próximo a 8000 milissegundos no experimento com 1500 solicitantes na solução construída com Java Socket.



(a) Tempo de resposta com Java Socket

(b) Tempo de resposta com Node.js

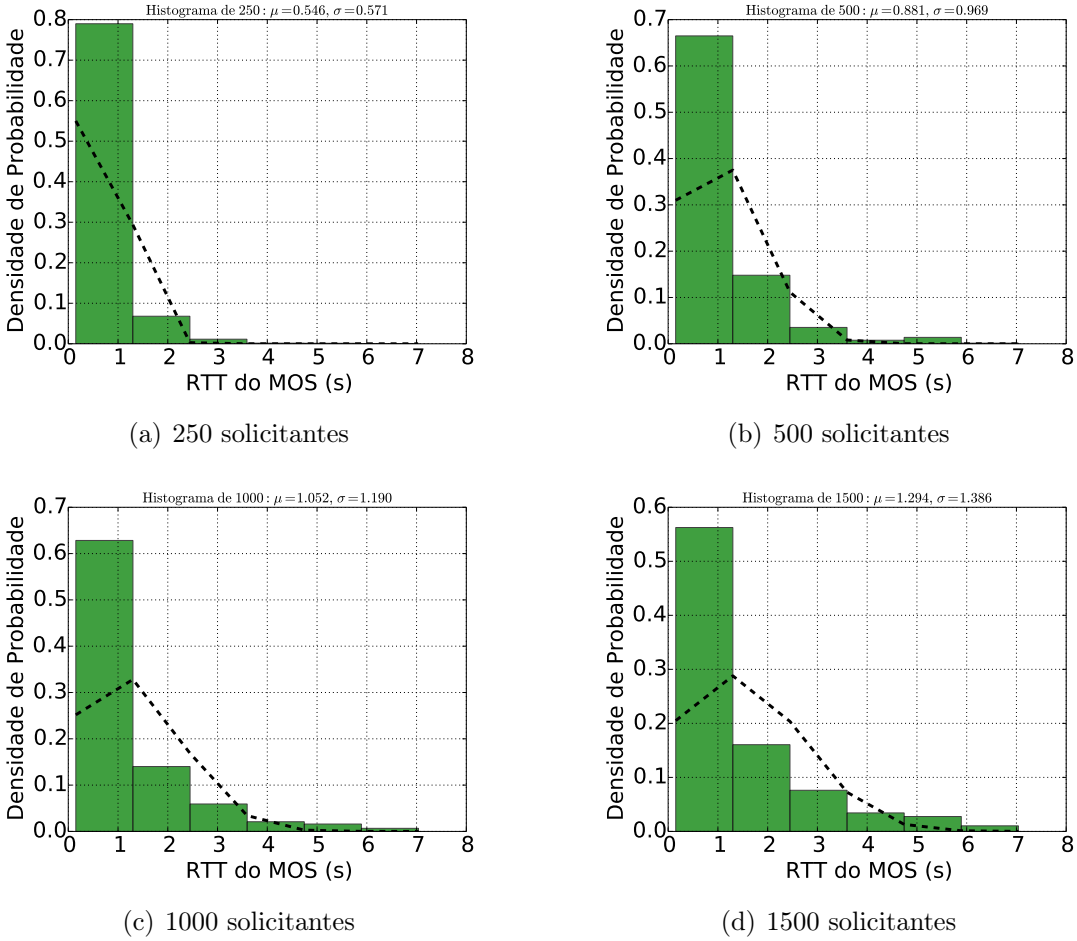


(c) Tempo médio de respostas positivas

(d) Média de respostas positivas

Figura 7.4: Avaliação das duas principais etapas de comunicação do MktPoints, o pedido de reserva e o registro do MOS

Os grupos apresentaram um pequeno aumento no coeficiente de variação, tanto em Java quanto no Node.js. Sendo 104,58%, 109,99%, 113,12% e 107,11%, para 250, 500, 1000 e 1500, respectivamente com Java. A queda no último percentual se deve aos problemas de memória enfrentados no servidor que discutiremos mais a frente e que nos levou ao desenvolvimento do SR em Node.js. Em ambos os casos, os histogramas mostraram um aumento linear na média e na dispersão conforme o número de solicitantes aumen-



(a) 250 solicitantes

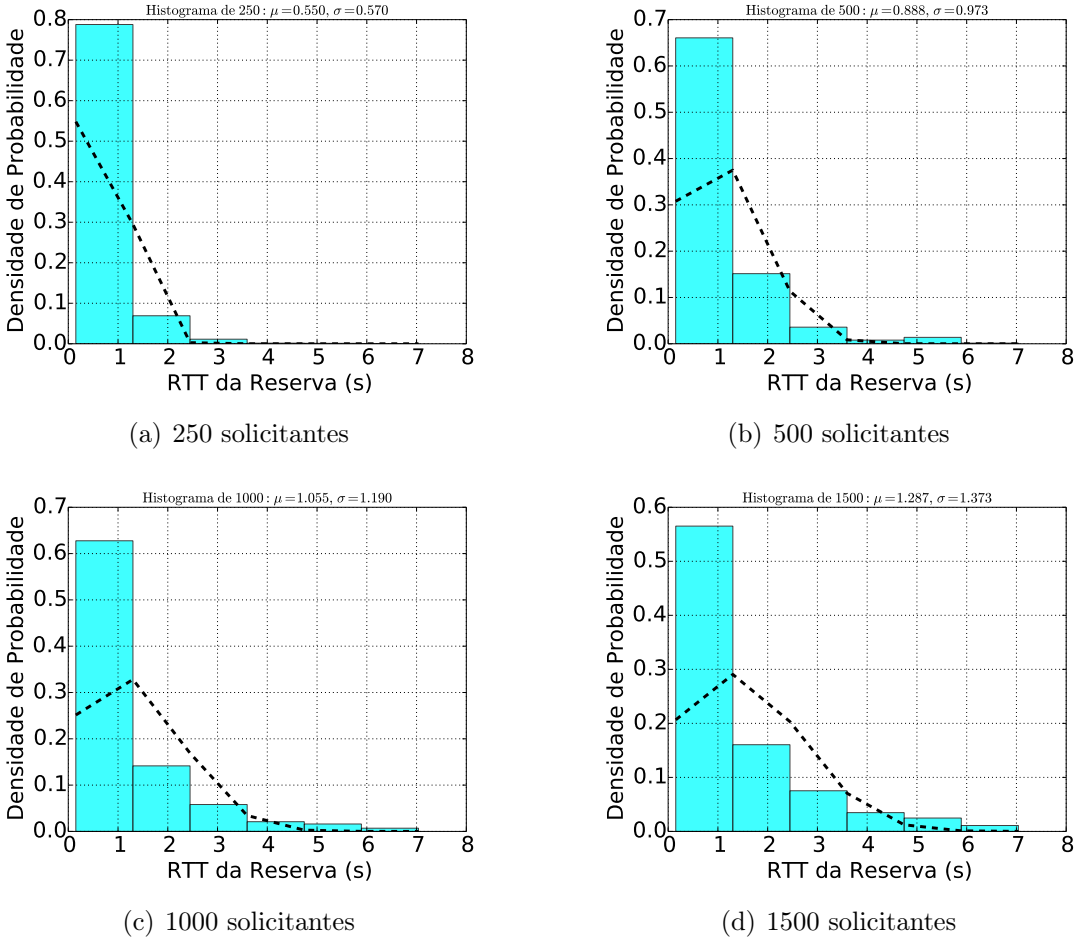
(b) 500 solicitantes

(c) 1000 solicitantes

(d) 1500 solicitantes

Figura 7.5: Deslocamento da média e aumento da variância no tempo de resposta (MOS com Java Socket)

tava. Esse fato já era esperado, visto que realizamos testes em um cenário semelhante ao modelo cliente-servidor. Já nos diagramas de caixa, devido à falta de simetria em todos os casos, a mediana se separa da média e aproxima-se do quartil inferior. O aumento da separação entre os quartis e os extremos demonstram o aumento da dispersão, por isso dispensamos a apresentação de todos os histogramas. A distância entre os quartis (desvio interquartílico) também aumenta e ilustra a faixa com 50% dos valores mais típicos da distribuição. Fora do intervalo entre os extremos estão os valores considerados discrepantes. Nas Figuras 7.4(a) e 7.4(b) já podemos ver uma diferença entre o SR com Java Socket e com Node.js: o aumento da latência é menor com Node.js. Outro detalhe que aparece é a distinção entre reserva e qualificação (MOS) com Node.js. Isso ocorre porque, diferente do Java que consegue realizar ambas as tarefas com uma única requisição ao banco de dados, o Node.js precisou de dois acessos ao banco para realizar a reserva e um para a qualificação. Logo, houve um reflexo na latência que fica mais evidente no experimento com 1500 solicitantes. A reserva é feita por meio de uma *Stored Procedure* que possui um



(a) 250 solicitantes

(b) 500 solicitantes

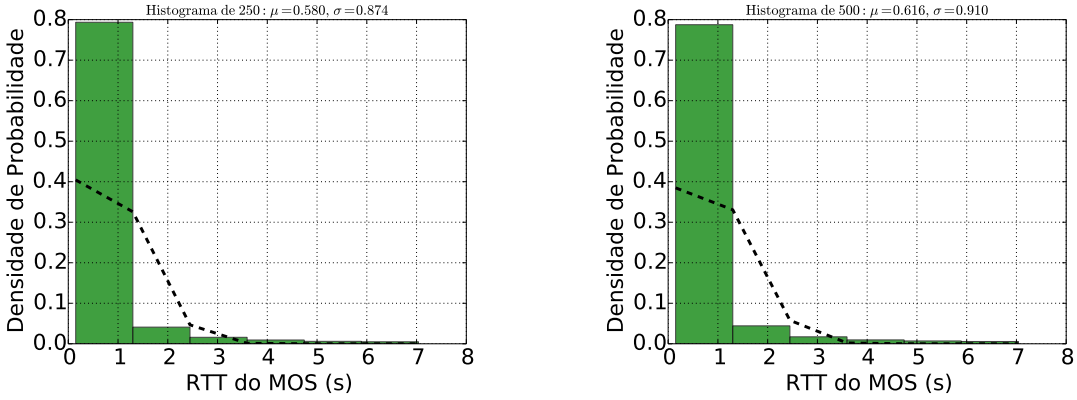
(c) 1000 solicitantes

(d) 1500 solicitantes

Figura 7.6: Deslocamento da média e aumento da variância no tempo de resposta (Reserva com Java Socket)

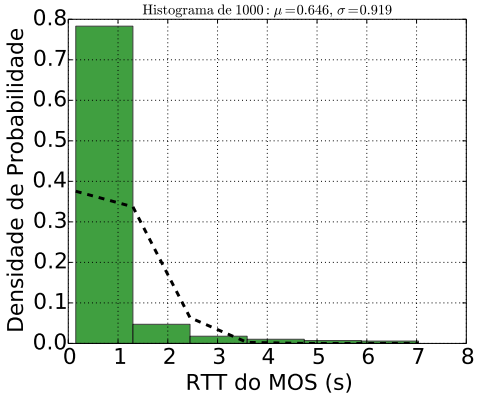
parâmetro *OUT* e o Node.js precisa realizar um *SELECT* nesse parâmetro após chamar esse procedimento.

O limite, representado na Figura 7.4(c), é calculado dividindo o tempo do experimento (10 minutos) por dois (uma solicitação de reserva e um MOS). Depois, o resultado é dividido pelo número de trabalhadores da campanha (200). Passar dessa linha, significa não ter tempo suficiente para mandar todas as 400 requisições (200 trabalhadores representam 200 pedidos de reserva e registros de MOS). Por isso, a coluna dos 1000 solicitantes na Figura 7.9 não tem o dobro do tamanho da coluna dos 500 no SR com Java Socket. Com 1000 solicitantes simultâneos, começamos a passar do limite na Figura 7.4(c) e isso afeta a quantidade de respostas. O SR com Node.js já passa do limite com 500, por isso a barra é um pouco menor que a do Java. Um dos fatores que pode ter contribuído para essa diferença é o protocolo HTTP usando pelo Node.js. Esse protocolo traz alguns benefícios, mas aumenta a quantidade de dados trafegado se comparado a um simples *socket*. A Figura 7.9 apresenta a quantidade média de todas as requisições

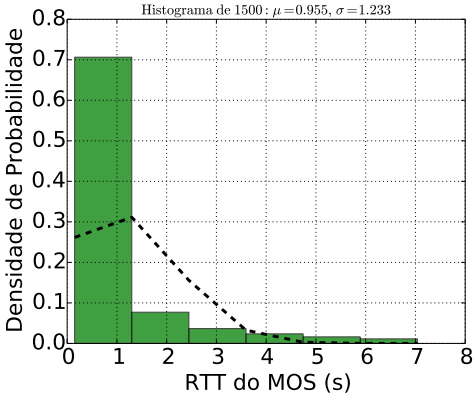


(a) 250 solicitantes

(b) 500 solicitantes



(c) 1000 solicitantes

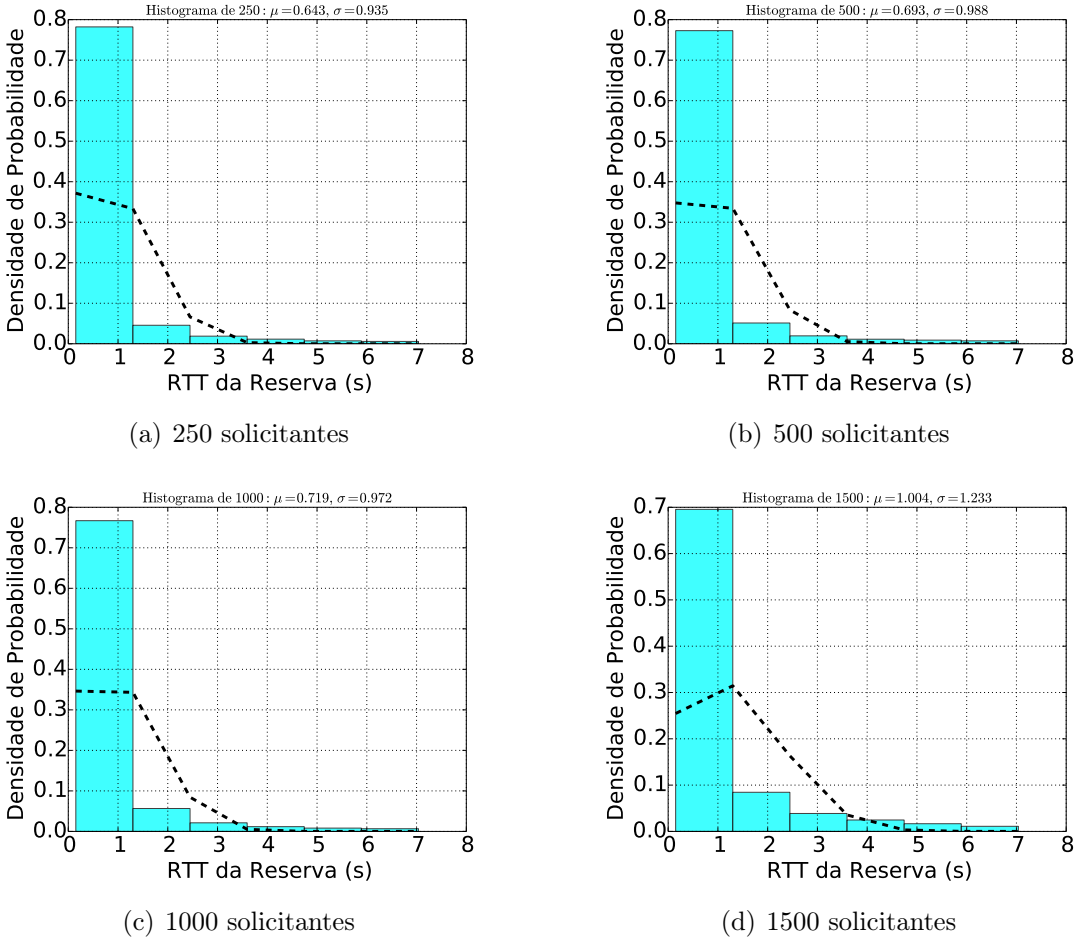


(d) 1500 solicitantes

Figura 7.7: Deslocamento da média e aumento da variância no tempo de resposta (MOS com Node.js)

entre as repetições do experimento, tanto de reserva quanto de MOS, com intervalo de confiança de 95% e apresentam os dados de todas as respostas positivas. As duas colunas referentes as respostas negativas não aparecem por representarem valores muito baixos, exceto nos experimentos com 1000 e 1500 solicitantes com Java Socket. As Figuras 7.4(c) e 7.4(d) também apresentam intervalo de confiança de 95%. Na Figura 7.4(d) o esperado é calculado multiplicando o número de trabalhadores (200) pelo número de solicitantes simultâneos. Vemos na Figura 7.4(d) um progressivo afastamento após exceder o limite na Figura 7.4(c). Um aumento linear e uma possível estagnação eram esperados (o que de fato ocorreu com o Node.js), mas não uma forte queda ao atingir 1500 solicitantes (o que aconteceu com o Java Socket). Fato também indicado na Figura 7.9.

As falhas no Java Socket com a concorrência de 1500 solicitantes ocorreu por questões de implementação do sistema. O experimento reservou 1,5 gigabytes de memória para o Java e limitou a alocação por *thread* em 228 kilobytes. Por conseguinte, ao receber 1500 requisições simultâneas tínhamos um consumo imediato de 334 megabytes. Após



(a) 250 solicitantes

(b) 500 solicitantes

(c) 1000 solicitantes

(d) 1500 solicitantes

Figura 7.8: Deslocamento da média e aumento da variância no tempo de resposta (Reserva com Node.js)

o atendimento dessas requisições a memória não era liberada imediatamente. Em Java, a liberação da memória é realizada por um recurso chamado Garbage Collection (GC). Mesmo chamando esse recurso diretamente, não há garantia de que a limpeza seja executada imediatamente. Caso o sistema tenha a capacidade de responder 1500 requisições em 500 milissegundos, após 2 segundos sem a execução do GC o serviço é interrompido por falta de memória. O *buffer* da rede começa a encher e mensagens negativas são disparadas, enquanto outras começam a apresentar atraso no processamento (começam a exceder o limite de 1500 ms). Buscando uma solução para esse problema chegamos ao Node.js[7] que tem a proposta de ser *single thread* e fornecer uma maneira fácil de criar programas de rede escaláveis. Ele também permite a criação de uma *thread* para cada núcleo de CPU e criação de *cluster*. O experimento utilizou um único núcleo de CPU e nesse ambiente simples o Node.js manteve-se estável ao trabalhar com 1500 solicitantes, conforme mostra a Figura 7.9. Os histogramas das Figuras 7.5 e 7.7 também mostram

[7]https://nodejs.org/

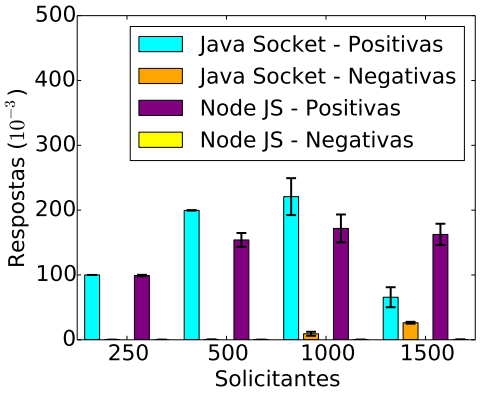


Figura 7.9: Avaliação da capacidade de atendimento das implementações com Java e NodeJS do MktPoints

que o Node.js responde a maior parte das requisições em pouco mais de 1 segundo, tanto de MOS quanto de reserva (Figuras 7.6 e 7.8).

7.3 Por que remover o atraso de rede?

O cenário apresentado na Seção 7.2.1 também foi adaptado para testar o Auspice. Porém, notamos que a quantidade de registros, a cada rodada, não passou de 60.000. Logo, ficou uma dúvida: o que aconteceu com as solicitações que não foram respondidas a tempo nesse experimento e no experimento com o SR/Node.js envolvendo os solicitantes (repare que chegamos a um platô na Figura 7.4(d))?

Após mais alguns testes foi possível perceber que o processo de leitura das chaves e assinatura dos comandos passados ao GNS causavam um atraso nas requisições. Logo, mesmo com o aumento dos trabalhadores o servidor encarregado de emular as requisições não conseguia montar mais de 60.000 a cada 10 minutos (tempo do experimento). O código passou por alterações para montagem de um lote de requisições fora do tempo do experimento, para posterior envio dentro do referido evento.

Os novos testes apontaram para uma limitação de memória, capacidade de processamento e número de portas abertas simultaneamente no emulador de usuários, indicando a necessidade de um número maior de máquinas trabalhando em paralelo.

Também não repetimos os testes com os solicitantes no SR, visto que o sistema não poderia responder mais de 30.000 requisições em 10 minutos (na teoria, como podemos ver na Seção 7.1.2) em um experimento que despreza o atraso de rede. Logo, uma emulação na Internet aumentaria os custos e não acrescentaria novas informações a esta tese.

Capítulo 8

Conclusão

Neste trabalho apresentamos uma nova arquitetura escalável para sistemas de Mobile Crowdsourcing e a primeira com suporte à mobilidade. A implementação da arquitetura, nomeada Snow Flurry, utiliza um serviço global de nomes, desenvolvido durante pesquisas ligadas à Internet do Futuro, e serviços em nuvem, para notificação dos trabalhadores e armazenamento das respostas. Realizamos com sucesso uma prova de conceito e discutimos diversas alternativas para eliminar o aporte financeiro adicional frente ao crescimento do fluxo de dados quando utilizamos a nuvem (apresentando, inclusive, uma solução para comunicação direta entre dispositivos). Os experimentos mostraram que existe um ganho entre 26,31% e 32,03% na redução da latência frente ao estado da arte. Além disso, uma análise derivada desses números mostrou um redução de pelo menos US$ 10,000.00 anuais nos custo e a dispensa de diversas atividades que resultavam em aumento da complexidade do sistema.

Também apresentamos e avaliamos a escalabilidade de um novo mecanismo de incentivo que permite a qualificação dos atendimentos, adiciona gamificação para prover motivação intrínseca, marketing de rede para aumentar o número de participantes e os princípios da caixa de Skinner para incentivar visitas ao sistema. Os resultados mostram que ele é capaz de atender a demanda do MTurk, um dos maiores *crowdsourcing* existentes. Mesmo com o ocorrido na implementação Java, devemos lembrar que o MTurk tem pouco mais de 1600 solicitantes ativos e a probabilidade de 90 ou mais solicitantes abrirem uma campanha em qualquer dia é de 0,007%.

8.1 Limitações

Por questões de escopo, tempo e viabilidade técnica, a arquitetura Snow Flurry carrega consigo algumas limitações, dentre as quais destacamos a aplicabilidade, visto que o Auspice está no centro da implementação do sistema e ele ainda se encontra na fase de "pesquisa e desenvolvimento". Outro fator importante é a questão da segurança, diversos autores relatam que os usuários relutam em participar de sistemas de MCS com receio de expor seus dados e localização atual. A arquitetura Snow Flurry não se aprofundou nesse tema, partindo do princípio que os usuários da plataforma não exigem anonimato. Por fim, outra grande limitação é o referencial com o estado da arte. Os trabalhos citados nesta tese realizaram simulações onde comparavam a solução proposta com variações da mesma ou com um esquema "padrão" desenvolvido pelos próprios autores. No geral, as soluções não se encontram disponíveis e os autores não respondem à *e-mails* solicitando informações para *download*. Também há casos em que, mesmo que a solução estivesse disponível, haveria a necessidade de superar custos e informações inconsistentes para um experimento em escala.

8.2 Trabalhos futuros

O *testbed* do ParticipAct (uma plataforma de *Mobile Crowd Sensing*), ocorreu na Universidade de Bolonha, Itália, e envolve 300 alunos por um ano em campanhas de sensoriamento passivo junto aos sensores de *smartphones* e também exigiu colaboração ativa do usuário [37]. Acreditamos que reproduzir um experimento, no universo do *Mobile Crowdsourcing*, pode revelar mais pontos sobre a operação de toda a arquitetura e fronteiras entre os dois paradigmas. O ParticipAct também envolveu dois pontos de interesse desta tese, uma proposta de mecanismo de incentivo [49] e um serviço de reputação [105] que utilizam redes sociais para atingir seus objetivos. Em Chessa et al. [49], os autores argumentam que um dos maiores desafios em um sistema MCS de longa duração reside na capacidade não apenas de atrair novos voluntários, mas também, e mais importante, de alavancar os laços sociais existentes entre os voluntários para mantê-los envolvidos na construção de comunidades MCS de longa duração. Kantarci, Glasser e Foschini [105] propuseram a utilização da teoria das redes sociais para avaliar a confiabilidade de dados *crowdsensed*, bem como os dispositivos móveis que fornecem serviços de detecção. Para isso, eles combinaram avaliações baseadas em reputação centralizadas com valores de reputação colaborativos baseados em votos e capacidades de voto. Cada participante foi

modelado como um nó em uma rede social onde os nós são interconectados através de seus valores de interação. Interação significa ter atribuído tarefas de detecção comum. Segundo os autores, esse procedimento melhora significativamente a utilidade da plataforma *crowdsensing*, reduzindo drasticamente a probabilidade de manipulação por nós maliciosos.

No que se refere ao mecanismo de incentivo, pretendemos compara-lo com o estado da arte, destacando quatro pontos: (1) racionalidade individual; (2) complexidade computacional; (3) verdade; e (4) bem-estar social. Também pretendemos avaliar a relevância de um processo automatizado para fixação de preços, estudar o arrependimento (*regret*) e buscar correlações estatísticas com outros métodos de pontuação da reputação.

Entre as questões técnicas, pretendemos: (1) utilizar o MSocket em Android e fazer uma análise comparativa com o NUTSS, sob a ótica dos firewalls, visto que o NUTSS alega prover *multi-homing* e mobilidade; (2) estudar a viabilidade de uma camada de segurança que promova o anonimato sem interferir na escalabilidade; (3) adicionar o tratamento de subtarefas; e (4) trabalho em equipe.

Referências

[1] Abdallaoui, H. E. A. E., Abdelaziz, E. F., Mohamed, S. Finding a lost child using a crowdsourcing framework. In *2016 4th International Conference on Control Engineering Information Technology (CEIT)* (2016), p. 1–6.

[2] Abdul-Rahman, A., Hailes, S. Supporting trust in virtual communities. In *Proceedings of the 33rd annual Hawaii international conference on system sciences* (2000), IEEE, p. 9–pp.

[3] Abelson, B., Varshney, K. R., Sun, J. Targeting direct cash transfers to the extremely poor. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14* (2014), p. 1563–1572.

[4] Abu-Elkheir, M., Hassanein, H. S., Oteafy, S. M. A. Enhancing emergency response systems through leveraging crowdsensing and heterogeneous data. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)* (2016), p. 188–193.

[5] Ahmed, N., Siddiquee, M. R., Karim, R., Zaman, M., Alam, S. M., Ashram, S. F. Crazy Crowd Sourcing to Mitigate Resource Scarcity. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (2017), p. 2322–2326.

[6] Akoh, B., Ahiabenu, K. A journey through 10 countries: Online election coverage in Africa. *Journalism Practice 6*, 3 (2012), 349–365.

[7] Al Noor, S., Hasan, R. D-cloc: A delay tolerant cloud formation using context-aware mobile crowdsourcing. In *Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on* (2015), IEEE, p. 147–154.

[8] Alfarrarjeh, A., Emrich, T., Shahabi, C. Scalable spatial crowdsourcing: A study of distributed algorithms. In *Mobile Data Management (MDM), 2015 16th IEEE International Conference on* (2015), vol. 1, IEEE, p. 134–144.

[9] Ali, A. U., Ogie, R. Social Media and Disasters: Highlighting Some Wicked Problems [Leading Edge]. *IEEE Technology and Society Magazine 36*, 4 (2017), 41–43.

[10] Ali, K., Al-Yaseen, D., Ejaz, A., Javed, T., Hassanein, H. S. CrowdITS: Crowdsourcing in intelligent transportation systems. In *IEEE Wireless Communications and Networking Conference, WCNC* (2012), p. 3307–3311.

[11] Aljadhai, A., Znati, T. F. Predictive mobility support for qos provisioning in mobile wireless environments. *Selected Areas in Communications, IEEE Journal on 19*, 10 (2001), 1915–1930.

[12] Allahbakhsh, M., Ignjatovic, A., Benatallah, B., Bertino, E., Foo, N., others. Reputation management in crowdsourcing systems. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on* (2012), IEEE, p. 664–671.

[13] Ally, A. Emerging Trends in Biosample-based Research: Mobile Technologies, Citizen Science, and Crowd-Sourcing Studies, 2014.

[14] Ambrosin, M., Compagno, A., Conti, M., Ghali, C., Tsudik, G. Security and privacy analysis of national science foundation future internet architectures. *IEEE Communications Surveys Tutorials 20*, 2 (Secondquarter 2018), 1418–1442.

[15] Amintoosi, H., Kanhere, S. S. A reputation framework for social participatory sensing systems. *Mobile Networks and Applications 19*, 1 (2014), 88–100.

[16] Amos, V. *Humanitarian response in the era of global mobile information technology*. Fordham University Press, 2013.

[17] Anchiêta, R. T., Moura, R. S. Exploring Unsupervised Learning Towards Extractive Summarization of User Reviews. In *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web* (New York, NY, USA, 2017), WebMedia '17, ACM, p. 217–220.

[18] Andersson, M., Sternberg, H. Informating Transport Transparency, 2016.

[19] Arbeláez, J. C., Osorio-Gómez, G. Crowdsourcing Augmented Reality Environment (CARE) for aesthetic evaluation of products in conceptual stage. *Computers in Industry 99* (2018), 241–252.

[20] Ardissono, L., Goy, A., Petrone, G., Segnan, M., Torasso, P. Intrigue: personalized recommendation of tourist attractions for desktop and hand held devices. *Applied artificial intelligence 17*, 8-9 (2003), 687–714.

[21] Ariffin, I., Solemon, B., Bakar, W. M. L. W. A. An evaluative study on mobile crowdsourcing applications for crime watch, 2014.

[22] Aris, H., Din, M. M. On using mobile crowdsourcing for timely information solicitation and sharing of prices, 2014.

[23] Aubry, E., Silverston, T., Lahmadi, A., Festor, O. CrowdOut: A mobile crowdsourcing service for road safety in digital cities, 2014.

[24] Banerjee, N., Wu, W., Das, S. K. Mobility support in wireless internet. *Wireless Communications, IEEE 10*, 5 (2003), 54–61.

[25] Barrón, J. P. G., Manso, M. Á., Alcarria, R., Gómez, R. P. A Mobile Crowdsourcing Platform for Urban Infrastructure Maintenance, 2014.

[26] Barroso, B. L. K., de Oliveira, R. R., Macedo, H. T. Mobile crowdsourcing app for smart cities, 2016.

[27] Baucke, S. Flexible architecture for the future internet. *7th Workshop ITG-5.2 "Future of Networks"* (2007). Acessado em 30/10/2015.

[28] Bilgic, M., Mooney, R. J. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization Workshop, IUI* (2005), vol. 5, p. 153.

[29] Billsus, D., Pazzani, M. J. A personal news agent that talks, learns and explains. In *Proceedings of the third annual conference on Autonomous Agents* (1999), Citeseer, p. 268–275.

[30] Bosker, B. The binge breaker. *The Atlantic* (2016).

[31] Bronzino, F., Nagaraja, K., Seskar, I., Raychaudhuri, D. Network service abstractions for a mobility-centric future internet architecture. In *Proceedings of the eighth ACM international workshop on Mobility in the evolving internet architecture* (2013), ACM, p. 5–10.

[32] Buaiz, S. Marketing de rede a fórmula da liderança: tudo o que você precisa saber para irradiar energia e confiança dentro de suas organizações, 1998.

[33] Callegati, F., Delnevo, G., Melis, A., Mirri, S., Prandini, M., Salomoni, P. I want to ride my bicycle: A microservice-based use case for a MaaS architecture. In *2017 IEEE Symposium on Computers and Communications (ISCC)* (2017), p. 18–22.

[34] Campista, M. E. M., Ferraz, L. H. G., Moraes, I. M., Lanza, M. L. D., Costa, L. H. M., Duarte, O. C. M. Interconexão de redes na internet do futuro: Desafios e soluções. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2010* (2010), 47–101.

[35] Capponi, A., Fiandrino, C., Kantarci, B., Foschini, L., Kliazovich, D., Bouvry, P. A survey on mobile crowdsensing systems: Challenges, solutions and opportunities. *IEEE Communications Surveys Tutorials* (2019), 1–1.

[36] Carbo, J., Molina, J. M., Davila, J. Trust management through fuzzy reputation. *International Journal of Cooperative Information Systems 12*, 01 (2003), 135–155.

[37] Cardone, G., Cirri, A., Corradi, A., Foschini, L. The participact mobile crowd sensing living lab: The testbed for smart cities. *IEEE Communications Magazine 52*, 10 (October 2014), 78–85.

[38] Carter, J., Bitting, E., Ghorbani, A. A. Reputation formalization for an information–sharing multi–agent system. *Computational Intelligence 18*, 4 (2002), 515–534.

[39] Castelfranchi, C., Falcone, R. Social trust: cognitive anatomy, social importance, quantification and dynamics. In *Proceedings of the First International Workshop on Trust* (1998), p. 35–49.

[40] Chang, Y. L., Liao, H. K., Shieh, C. K., Miao, Y. B. KaiKai: A NAT traversal approach by using protocol behavior analysis. In *Proceedings - 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2006* (2006), p. 177–180.

[41] Chen, B., Zhou, W., Li, Y., Guo, D. Innovative application of SIP protocol for communication platform. In *Proceedings - 2010 International Conference on Anti-Counterfeiting, Security and Identification, 2010 ASID* (2010), p. 323–326.

[42] Chen, S., Li, M., Ren, K., Fu, X., Qiao, C. Rise of the Indoor Crowd: Reconstruction of Building Interior View via Mobile Crowdsourcing. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems* (New York, NY, USA, 2015), SenSys '15, ACM, p. 59–71.

[43] Chen, W. E., Chen, B. E. An Effective NAT Traversal Mechanism for SIP/IMS Services in SDN-Enabled All-IP Mobile Networks. *Wireless Personal Communications 84*, 3 (2015), 2171–2185.

[44] Chen, W. E., Huang, Y. L., Chao, H. C. NAT traversing solutions for SIP applications. *Eurasip Journal on Wireless Communications and Networking 2008* (2008).

[45] Chen, X., Santos-Neto, E., Ripeanu, M. Crowdsourcing for On-street Smart Parking. In *Proceedings of the Second ACM International Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications* (New York, NY, USA, 2012), DIVANet '12, ACM, p. 1–8.

[46] Chen, Y. C., Jia, W. K. Challenge and solutions of NAT traversal for ubiquitous and pervasive applications on the Internet. *Journal of Systems and Software 82*, 10 (2009), 1620–1626.

[47] Chen, Z., Fu, R., Zhao, Z., Liu, Z., Xia, L., Chen, L., Cheng, P., Cao, C. C., Tong, Y., Zhang, C. J. gmission: A general spatial crowdsourcing platform. *Proceedings of the VLDB Endowment 7*, 13 (2014), 1629–1632.

[48] Cheng, P., Lian, X., Chen, Z., Fu, R., Chen, L., Han, J., Zhao, J. Reliable diversity-based spatial crowdsourcing by moving workers. *Proceedings of the VLDB Endowment 8*, 10 (2015), 1022–1033.

[49] Chessa, S., Corradi, A., Foschini, L., Girolami, M. Empowering mobile crowdsensing through social and ad hoc networking. *IEEE Communications Magazine 54*, 7 (July 2016), 108–114.

[50] Christin, D., Reinhardt, A., Kanhere, S. S., Hollick, M. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software 84*, 11 (2011), 1928–1946.

[51] Conitzer, V., Immorlica, N., Letchford, J., Munagala, K., Wagman, L. False-name-proofness in social networks. In *International Workshop on Internet and Network Economics* (2010), Springer, p. 209–221.

[52] Cosley, D., Lam, S. K., Albert, I., Konstan, J. A., Riedl, J. Is seeing believing?: how recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2003), ACM, p. 585–592.

[53] Costa, L. A. O sistema de marketing de rede: uma estratégia de ação mercadológica, Janeiro 2001.

[54] Coughlan, A. T., Grayson, K. Network marketing organizations: Compensation plans, retail network growth, and profitability. *International Journal of Research in Marketing 15*, 5 (1998), 401–426.

[55] Croft, R., Woodruffe, H. Network marketing: The ultimate in international distribution? *Journal of Marketing Management 12*, 1-3 (1996), 201–214.

[56] Cuevas, R., Cuevas, Á., Cabellos-Aparicio, A., Jakab, L., Guerrero, C. A collaborative P2P scheme for NAT Traversal Server discovery based on topological information. *Computer Networks 54*, 12 (2010), 2071–2085.

[57] Czarkowski, M., Kay, J. A scrutable adaptive hypertext. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* (2002), Springer, p. 384–387.

[58] Danezis, G., Lewis, S., Anderson, R. J. How much is location privacy worth? In *WEIS* (2005), vol. 5, Citeseer.

[59] Dang, H., Nguyen, T., To, H. Maximum complex task assignment: Towards tasks correlation in spatial crowdsourcing. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services* (2013), ACM, p. 77.

[60] D'Angelo, G., Rampone, S. A NAT traversal mechanism for cloud video surveillance applications using WebSocket, 2018.

[61] de Brito, G. M., Velloso, P. B., Moraes, I. M. Redes orientadas a conteúdo: Um novo paradigma para a internet. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012* (2012), 211–264.

[62] de Brito, G. M., Velloso, P. B., Moraes, I. M. Uma análise do desempenho de redes sem-fio orientadas a conteudo. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)* (2014).

[63] Deng, D., Shahabi, C., Demiryurek, U. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *Proceedings of the 21st acm sigspatial international conference on advances in geographic information systems* (2013), ACM, p. 324–333.

[64] Deng, D., Shahabi, C., Zhu, L. Task Matching and Scheduling for Multiple Workers in Spatial Crowdsourcing. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems* (New York, NY, USA, 2015), SIGSPATIAL '15, ACM, p. 21:1—-21:10.

[65] Ding, C., Cheng, H. K., Duan, Y., Jin, Y. The power of the "like" button: The impact of social media on box office. *Decision Support Systems 94* (2017), 77–84.

[66] Douceur, J. R., Moscibroda, T. Lottery trees: motivational deployment of networked systems. *ACM SIGCOMM Computer Communication Review 37*, 4 (2007), 121–132.

[67] Drucker, F. A., Fleischer, L. K. Simpler sybil-proof mechanisms for multi-level marketing. In *Proceedings of the 13th ACM conference on Electronic commerce* (2012), ACM, p. 441–458.

[68] Duan, L., Kubo, T., Sugiyama, K., Huang, J., Hasegawa, T., Walrand, J. Incentive mechanisms for smartphone collaboration in data acquisition and distributed computing. In *2012 Proceedings IEEE INFOCOM* (2012), IEEE, p. 1701–1709.

[69] Emek, Y., Karidi, R., Tennenholtz, M., Zohar, A. Mechanisms for multi-level marketing. In *Proceedings of the 12th ACM conference on Electronic commerce* (2011), ACM, p. 209–218.

[70] Eppinger, J. TCP connections for P2P apps: A software approach to solving the NAT problem. *Institute for Software Research* (2005), –05–104.

[71] Esfandiari, B., Chandrasekharan, S. On how agents make friends: Mechanisms for trust acquisition. In *Proceedings of the fourth workshop on deception, fraud and trust in agent societies* (2001), vol. 222, p. 19.

[72] Eyal, N. *Hooked: How to build habit-forming products*. Penguin UK, 2014.

[73] Falcone, R., Castelfranchi, C. Social trust: A cognitive approach. In *Trust and deception in virtual societies*. Springer, 2001, p. 55–90.

[74] Feng, W., Yan, Z., Zhang, H., Zeng, K., Xiao, Y., Hou, Y. T. A survey on security, privacy, and trust in mobile crowdsourcing. *IEEE Internet of Things Journal 5*, 4 (Aug 2018), 2971–2992.

[75] Feng, Z., Zhu, Y., Zhang, Q., Ni, L. M., Vasilakos, A. V. Trac: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications* (2014), IEEE, p. 1231–1239.

[76] Foundation, O. S. M. Open street map, 2016. Disponível em `http://www.openstreetmap.org/copyright`.

[77] Ganti, R. K., Ye, F., Lei, H. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine 49*, 11 (2011), 32–39.

[78] Garcin, F., Faltings, B., Jurca, R. Aggregating reputation feedback. *Proceedings of the First International Conference on Reputation: Theory and Technology 1*, 1 (2009), 62–74.

[79] Gaylani, N., Erten, Y. M. Handling NAT traversal and mobility for multimedia traffic. In *2006 3rd IEEE Consumer Communications and Networking Conference, CCNC 2006* (2006), vol. 1, p. 112–116.

[80] GNS. Msocket documentation, oct 2015.

[81] Gracioso, F., Najjar, E. R. Marketing de rede: a era do supermercado virtual, 1997.

[82] Guha, S., Francis, P. Characterization and measurement of TCP traversal through NATs and firewalls. In *Proceedings of the 5th ACM SIGCOMM conference on Internet measurement - IMC '05* (2005), p. 1.

[83] Guha, S., Francis, P. An end-middle-end approach to connection establishment. *Sigcomm* (2007), 12.

[84] Han, Y., Wu, H. Minimum-cost crowdsourcing with coverage guarantee in mobile opportunistic d2d networks. *IEEE Transactions on Mobile Computing* (2017).

[85] Hara, T., Springer, T., Bombach, G., Schill, A. Decentralised Approach for a Reusable Crowdsourcing Platform Utilising Standard Web Servers. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication* (New York, NY, USA, 2013), UbiComp '13 Adjunct, ACM, p. 1063–1074.

[86] Herlocker, J. L., Konstan, J. A., Riedl, J. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (2000), ACM, p. 241–250.

[87] Herzig, A., Lorini, E., Hübner, J. F., Ben-Naim, J., Castelfranchi, C., Demolombe, R., Longin, D., Vercouter, L., Boissier, O. Prolegomena for a logic of trust and reputation. *NorMAS 8* (2008), 143–157.

[88] Hoffman, K., Zage, D., Nita-Rotaru, C. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR) 42*, 1 (2009), 1.

[89] Howe, J. The rise of crowdsourcing. *Wired magazine 14*, 6 (2006), 1–4.

[90] Huynh, T. D., Jennings, N. R., Shadbolt, N. R. An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems 13*, 2 (2006), 119–154.

[91] Jagabathula, S., Subramanian, L., Venkataraman, A. Reputation-based worker filtering in crowdsourcing. In *Advances in Neural Information Processing Systems* (2014), p. 2492–2500.

[92] Jaimes, L. G., Vergara-Laurens, I., Labrador, M. A. A location-based incentive mechanism for participatory sensing systems with budget constraints. In *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on* (2012), IEEE, p. 103–108.

[93] Jaimes, L. G., Vergara-Laurens, I. J., Raij, A. A survey of incentive techniques for mobile crowd sensing. *IEEE Internet of Things Journal 2*, 5 (2015), 370–380.

[94] Jain, S., Narayanaswamy, B., Narahari, Y. A multiarmed bandit incentive mechanism for crowdsourcing demand response in smart grids. In *AAAI* (2014), p. 721–727.

[95] Jarvis, C. The rise and fall of the pyramid schemes in albania. *IMF staff papers 47*, 1 (2000), 1–29.

[96] Jin, H., Su, L., Chen, D., Nahrstedt, K., Xu, J. Quality of information aware incentive mechanisms for mobile crowd sensing systems. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing* (2015), ACM, p. 167–176.

[97] Johnson, D., Perkins, C., Arkko, J. Mobility support in ipv6. Relatório Técnico, The Internet Society, 2004.

[98] Jøsang, A., Ismail, R., Boyd, C. A survey of trust and reputation systems for online service provision. *Decision support systems 43*, 2 (2007), 618–644.

[99] JS, L. Leaflet an open-source javascript library for mobile-friendly interactive maps, 2016. Disponível em `http://leafletjs.com/`.

[100] Kamar, E., Horvitz, E. Incentives for truthful reporting in crowdsourcing. In *Proceedings of the 11th international conference on autonomous agents and multiagent systems-volume 3* (2012), International Foundation for Autonomous Agents and Multiagent Systems, p. 1329–1330.

[101] Kamienski, C., Oliveira Biondi, G., Borelli, F. F., Heideker, A., Ratusznei, J., Kleinschmidt, J. H. Capítulo 2 Computação Urbana: Tecnologias e Aplicações para Cidades Inteligentes. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos* (2016).

[102] Kandappu, T., Misra, A., Cheng, S.-F., Jaiman, N., Tandriansyah, R., Chen, C., Lau, H. C., Chander, D., Dasgupta, K. Campus-Scale Mobile Crowd-Tasking: Deployment & Behavioral Insights. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing* (New York, NY, USA, 2016), CSCW '16, ACM, p. 800–812.

[103] Kanhere, S. S. Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In *2011 IEEE 12th International Conference on Mobile Data Management* (2011), vol. 2, IEEE, p. 3–6.

[104] Kanhere, S. S. Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In *International Conference on Distributed Computing and Internet Technology* (2013), Springer, p. 19–26.

[105] Kantarci, B., Glasser, P. M., Foschini, L. Crowdsensing with social network-aided collaborative trust scores. In *2015 IEEE Global Communications Conference (GLOBECOM)* (2015), IEEE, p. 1–6.

[106] Kazemi, L., Shahabi, C. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems* (2012), ACM, p. 189–198.

[107] Kazemi, L., Shahabi, C., Chen, L. Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In *Proceedings of the 21st acm sigspatial international conference on advances in geographic information systems* (2013), ACM, p. 314–323.

[108] Konert, J., Söbke, H., Wendel, V. Social network games. In *Entertainment Computing and Serious Games*. Springer, 2016, p. 442–474.

[109] Kong, K.-S., Lee, W., Han, Y.-H., Shin, M.-K., You, H. Mobility management for all-ip mobile networks: mobile ipv6 vs. proxy mobile ipv6. *Wireless Communications, IEEE 15*, 2 (2008), 36–45.

[110] Koutsopoulos, I. Optimal incentive-driven design of participatory sensing systems. In *Infocom, 2013 proceedings ieee* (2013), IEEE, p. 1402–1410.

[111] Krontiris, I., Albers, A. Monetary incentives in participatory sensing using multi-attributive auctions. *International Journal of Parallel, Emergent and Distributed Systems 27*, 4 (2012), 317–336.

[112] Lee, J.-S., Hoh, B. Dynamic pricing incentive for participatory sensing. *Pervasive and Mobile Computing 6*, 6 (2010), 693–708.

[113] Levkowetz, H., Vaarala, S. Mobile ip traversal of network address translation (nat) devices. Relatório Técnico, The Internet Society, 2003.

[114] Li, J., He, L.-w. Automated nat traversal for peer-to-peer networks, mar 2011. US Patent 7,912,046.

[115] Li, M., Weng, J., Yang, A., Lu, W., Zhang, Y., Hou, L., Liu, J., Xiang, Y., Deng, R. H. Crowdbc: A blockchain-based decentralized framework for crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems 30*, 6 (June 2019), 1251–1266.

[116] Liu, M., Liu, B., Liu, J., Du, M. A novel solution based on NAT traversal for high-speed accessing the campus network from the public network. *Research Journal of Applied Sciences, Engineering and Technology 7*, 2 (2014), 221–226.

[117] Liu, S., Jin, H., Liao, X., Yao, H., Zeng, D. TCPBridge: A software approach to establish direct communications for NAT hosts. In *AICCSA 08 - 6th IEEE/ACS International Conference on Computer Systems and Applications* (2008), p. 247–252.

[118] Luo, Y., Eymann, J., Timm-Giel, A. Mobility support for content centric networking. *Telecommunication Systems 59*, 2 (2015), 271–288.

[119] Lv, Y., Moscibroda, T. Fair and resilient incentive tree mechanisms. *Distributed Computing 29*, 1 (2016), 1–16.

[120] Marjanović, M., Antonić, A., Žarko, I. P. Edge computing architecture for mobile crowdsensing. *IEEE Access 6* (2018), 10662–10674.

[121] Marsh, S. P. *Formalising trust as a computational concept.* Tese de Doutorado, University of Stirling, 1994.

[122] Mc Mahon, C. Why do we'like'social media? *Psychologist 28*, 9 (2015), 724–728.

[123] McCarthy, K., Reilly, J., McGinty, L., Smyth, B. Thinking positively-explanatory feedback for conversational recommender systems. In *Proceedings of the European Conference on Case-Based Reasoning (ECCBR-04) Explanation Workshop* (2004), p. 115–124.

[124] McGinty, L., Smyth, B. Extending comparison-based recommendation: A review. *Poster acceptance for the British Computer Society's Specialist Group on Artificial Intelligence (AI-03)* (2003).

[125] McLeod, S. Skinner-operant conditioning. *Retrieved from* (2015).

[126] McSherry, D. Explanation in recommender systems. *Artificial Intelligence Review 24*, 2 (2005), 179–197.

[127] Milgram, S. The small world problem. *Psychology today 2*, 1 (1967), 60–67.

[128] Moore, A. L. Building a successful network marketing company: the systems, the products, and the know-how you need to launch or enhance a successful mlm company, 1998.

[129] Mui, L., Mohtashemi, M., Halberstadt, A. A computational model of trust and reputation. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences* (2002), IEEE, p. 2431–2439.

[130] Mukherjee, S., Baid, A., Raychaudhuri, D. Integrating advanced mobility services into the future internet architecture. *7th International Conference on Communication Systems and Networks (COMSNETS)* (2015).

[131] Muller, G., Vercouter, L. Decentralized monitoring of agent communications with a reputation model. In *Trusting Agents for Trusting Electronic Societies*. Springer, 2004, p. 144–161.

[132] Muller, W., Silva, T. H., Almeida, J. M., Loureiro, A. A. F. Study of Gender Preferences for Locations Around the World Using Social Media Data. In *Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web* (New York, NY, USA, 2016), Webmedia '16, ACM, p. 295–302.

[133] Musthag, M., Raij, A., Ganesan, D., Kumar, S., Shiffman, S. Exploring micro-incentive strategies for participant compensation in high-burden studies. In *Proceedings of the 13th international conference on Ubiquitous computing* (2011), ACM, p. 435–444.

[134] Nath, S., Dayama, P., Garg, D., Narahari, Y., Zou, J. Mechanism design for time critical and cost critical task execution via crowdsourcing. In *International Workshop on Internet and Network Economics* (2012), Springer, p. 212–226.

[135] Neyman, C. J. A survey of addictive software design. Relatório Técnico, Digital Commons @ Cal Poly, 2017.

[136] Padovan, B., Sackmann, S., Eymann, T., Pippow, I. A prototype for an agent-based secure electronic marketplace including reputation-tracking mechanisms. *International Journal of Electronic Commerce 6*, 4 (2002), 93–113.

[137] Peng, X., Gu, J., Tan, T. H., Sun, J., Yu, Y., Nuseibeh, B., Zhao, W. Crowdservice: serving the individuals through mobile crowdsourcing and service composition. In *Automated Software Engineering (ASE), 2016 31st IEEE/ACM International Conference on* (2016), IEEE, p. 214–219.

[138] Perkins, C. Ip mobility support for ipv4, revised. Relatório Técnico, Internet Engineering Task Force (IETF), 2010.

[139] Phuttharak, J., Loke, S. W. A review of mobile crowdsourcing architectures and challenges: Toward crowd-empowered internet-of-things. *IEEE Access 7* (2019), 304–324.

[140] Pickard, G., Pan, W., Rahwan, I., Cebrian, M., Crane, R., Madan, A., Pentland, A. Time-critical social mobilization. *Science 334*, 6055 (2011), 509–512.

[141] Pinyol, I., Sabater-Mir, J. Pragmatic-strategic reputation-based decisions in bdi agents. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2* (2009), International Foundation for Autonomous Agents and Multiagent Systems, p. 1001–1008.

[142] Pinyol, I., Sabater-Mir, J. Computational trust and reputation models for open multi-agent systems: a review. *Artificial Intelligence Review 40*, 1 (2013), 1–25.

[143] Pinyol, I., Sabater-Mir, J., Dellunde, P., Paolucci, M. Reputation-based decisions for logic-based cognitive agents. *Autonomous Agents and Multi-Agent Systems 24*, 1 (2012), 175–216.

[144] Pu, P., Chen, L. Trust building with explanation interfaces. In *Proceedings of the 11th international conference on Intelligent user interfaces* (2006), ACM, p. 93–100.

[145] Ra, M.-R., Liu, B., La Porta, T. F., Govindan, R. Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (2012), ACM, p. 337–350.

[146] Rasmusson, L., Jansson, S. Simulated social control for secure internet commerce (position paper). In *Proceedings, New Security Paradigms Workshop, Lake Arrowhead* (1996).

[147] Rasmusson, L., Rasmusson, A., Janson, S. Using agents to secure the internet marketplace reactive security and social control. In *Proceedings of PAAM* (1997), vol. 97, p. 41.

[148] Raychaudhuri, D., Nagaraja, K., Venkataramani, A. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGMOBILE Mobile Computing and Communications Review 16*, 3 (2012), 2–13.

[149] Raza, M. A., Ahmed, R., Boutaba, R. URL forwarding for NAT traversal. In *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015* (2015), p. 599–605.

[150] Reddy, S., Estrin, D., Hansen, M., Srivastava, M. Examining micropayments for participatory sensing data collections. In *Proceedings of the 12th ACM international conference on Ubiquitous computing* (2010), ACM, p. 33–36.

[151] Regan, K., Cohen, R. Indirect reputation assessment for adaptive buying agents in electronic markets. In *Business Agents and the Semantic Web workshop* (2005), vol. 1.

[152] Ren, J., Zhang, Y., Zhang, K., Shen, X. S. Sacrm: social aware crowdsourcing with reputation management in mobile sensing. *Computer Communications 65* (2015), 55–65.

[153] Ribeiro, F., Florêncio, D., Zhang, C., Seltzer, M. Crowdmos: An approach for crowdsourcing mean opinion score studies. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (2011), IEEE, p. 2416–2419.

[154] Ripperger, T. *Ökonomik des Vertrauens–Analyse eines Organisationsprinzips Tbinger*. Mohr Siebeck, 1998.

[155] Rosenberg, J. Interactive connectivity establishment (ice): A protocol for network address translator (nat) traversal for offer/answer protocols. Relatório Técnico, Internet Engineering Task Force (IETF), 2010.

[156] Sabater, J., Paolucci, M., Conte, R. Repage: Reputation and image among limited autonomous partners. *Journal of artificial societies and social simulation 9*, 2 (2006).

[157] Sabater, J., Sierra, C. Regret: reputation in gregarious societies. In *Agents* (2001), vol. 1, p. 194–195.

[158] Sabater, J., Sierra, C. Reputation and social network analysis in multi-agent systems. In *AAMAS* (2002), vol. 2, p. 475–482.

[159] Sabater, J., Sierra, C. Review on computational trust and reputation models. *Artificial intelligence review 24*, 1 (2005), 33–60.

[160] Sabater i Mir, J. *Trust and reputation for agent societies*. Universitat Autònoma de Barcelona,, 2004.

[161] Schillo, M., Funk, P., Rovatsos, M. Who can you trust: Dealing with deception. In *Proceedings of the workshop Deception, Fraud and trust in Agent Societies at the Autonomous Agents Conference* (1999), p. 95–106.

[162] Schillo, M., Funk, P., Rovatsos, M. Using trust for detecting deceitful agents in artificial societies. *Applied Artificial Intelligence 14*, 8 (2000), 825–848.

[163] Schwartz, C., Borchert, K., Hirth, M., Tran-Gia, P. Modeling crowdsourcing platforms to enable workforce dimensioning. In *Telecommunication Networks and Applications Conference (ITNAC), 2015 International* (2015), IEEE, p. 30–37.

[164] Sen, S., Sajja, N. Robustness of reputation-based trust: Boolean case. In *Proceedings of the first international joint conference on Autonomous agents and multi-agent systems: part 1* (2002), ACM, p. 288–293.

[165] Shahabi, C. Towards a generic framework for trustworthy spatial crowdsourcing. In *Proceedings of the 12th International ACM Workshop on Data Engineering for Wireless and Mobile Acess* (2013), ACM, p. 1–4.

[166] Sharma, A., Tie, X., Uppal, H., Venkataramani, A., Westbrook, D., Yadav, A. A global name service for a highly mobile internetwork. *ACM SIGCOMM Computer Communication Review 44*, 4 (2014), 247–258.

[167] Sharma, A., Tie, X., Uppal, H., Venkataramani, A., Westbrook, D., Yadav, A. A global name service for a highly mobile internetwork. *ACM SIGCOMM Computer Communication Review 44*, 4 (2015), 247–258.

[168] Sierra, C., Debenham, J. K. An information-based model for trust. In *International Conference on Autonomous Agents and Multiagent Systems* (2005), ACM.

[169] Silva, T. H., de Melo, P. O. V., Neto, J., IJT, A., Ribeiro, C. S. d. S., Mota, V., da Cunha, F. D., Ferreira, A., Machado, K. L. d. S., Mini, R. A. d. F., others. Redes de sensoriamento participativo: Desafios e oportunidades. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos* (2015).

[170] Singer, Y., Mittal, M. Pricing mechanisms for crowdsourcing markets. In *Proceedings of the 22nd international conference on World Wide Web* (2013), ACM, p. 1157–1166.

[171] Singla, A., Krause, A. Incentives for privacy tradeoff in community sensing. In *First AAAI Conference on Human Computation and Crowdsourcing* (2013).

[172] Singla, A., Krause, A. Truthful incentives in crowdsourcing tasks using regret minimization mechanisms. In *Proceedings of the 22nd international conference on World Wide Web* (2013), ACM, p. 1167–1178.

[173] Sinha, R., Swearingen, K. The role of transparency in recommender systems. In *CHI'02 extended abstracts on Human factors in computing systems* (2002), ACM, p. 830–831.

[174] Söllner, M., Görg, C., Pentikousis, K., Lopez, J. M. C., de Leon, M. P., Bertin, P. Mobility scenarios for the future internet: The 4ward approach. *Em 11th International Symposium on Wireless Personal Multimedia Communications (WPMC 2008)* (2008).

[175] Subramanian, A., Kanth, G. S., Vaze, R. Offline and online incentive mechanism design for smart-phone crowd-sourcing. *arXiv preprint arXiv:1310.1746* (2013).

[176] Swartout, W. R. Rule-based expert systems: The mycin experiments of the stanford heuristic programming project, 1985.

[177] Tanaka, H., Suzuki, H., Watanabe, A., Naito, K. Proposal for a secure end-to-end remote control system for ECHONET lite home appliances. In *2017 IEEE 6th Global Conference on Consumer Electronics, GCCE 2017* (2017), vol. 2017-Janua, p. 1–2.

[178] Tanaka, H., Suzuki, H., Watanabe, A., Naito, K. Evaluation of a secure end-to-end remote control system for smart home appliances. In *2018 IEEE International Conference on Consumer Electronics (ICCE)* (jan 2018), p. 1–2.

[179] Tanenbaum, A. *Rede de Computadores. 3 edição.* Editora Campus, 2006.

[180] Thompson, C. A., Goker, M. H., Langley, P. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research 21* (2004), 393–428.

[181] Tie, X., Sharma, A., Venkataramani, A. A global name service for a highly mobile internet. Relatório Técnico, UMASS Computer Science Technical Report: UM-CS-2013, 2013.

[182] Tintarev, N., Masthoff, J. A survey of explanations in recommender systems. In *2007 IEEE 23rd international conference on data engineering workshop* (2007), IEEE, p. 801–810.

[183] To, H., Ghinita, G., Shahabi, C. A framework for protecting worker location privacy in spatial crowdsourcing. *Proceedings of the VLDB Endowment 7*, 10 (2014), 919–930.

[184] To, H., Shahabi, C., Kazemi, L. A server-assigned spatial crowdsourcing framework. *ACM Transactions on Spatial Algorithms and Systems 1*, 1 (2015), 2.

[185] Vander Nat, P. J., Keep, W. W. Marketing Fraud: An Approach for Differentiating Multilevel Marketing from Pyramid Schemes. *Journal of Public Policy & Marketing 21*, 1 (2002), 139–151.

[186] Venkataramani, A., Sharma, A., Tie, X., Uppal, H., Westbrook, D., Kurose, J., Raychaudhuri, D. Design requirements of a global name service for a mobility-centric, trustworthy internetwork. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on* (2013), IEEE, p. 1–9.

[187] Venkataramni, A., Gao, Z., Gu, T., Anantharamu, K. Scalable fine-grained reconfigurable replica coordination. Relatório Técnico, University of Massachusetts (UMass), 2018.

[188] von Ahn, L. *Human Computation.* Tese de Doutorado, Carnegie Mellon University, dec 2005.

[189] Von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M. recaptcha: Human-based character recognition via web security measures. *Science 321*, 5895 (2008), 1465–1468.

[190] Wang, K., Gu, L., Guo, S., Chen, H., Leung, V. C., Sun, Y. Crowdsourcing-based content-centric network: a social perspective. *IEEE network 31*, 5 (2017), 28–34.

[191] Wang, M., Yan, Z. Security in D2D communications: A review. *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015 1* (2015), 1199–1204.

[192] Wang, Y., Bi, J., Zhang, K. Design and implementation of a software-defined mobility architecture for ip networks. *Mobile Networks and Applications 20*, 1 (2015), 40–52.

[193] Wang, Y., Cai, Z., Zhan, Z., Gong, Y., Tong, X. An optimization and auction-based incentive mechanism to maximize social welfare for mobile crowdsourcing. *IEEE Transactions on Computational Social Systems* (2019), 1–16.

[194] Wärnestål, P. User evaluation of a conversational recommender system. In *Proceedings of the 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems* (2005), p. 32–39.

[195] Willkie, J. J., Lioy, M., Armstrong, S. M. Ip mobility support using proxy mobile node registration, may 2001. US Patent 6,230,012.

[196] Xiao, Y., Simoens, P., Pillai, P., Ha, K., Satyanarayanan, M. Lowering the barriers to large-scale mobile crowdsensing. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications* (2013), ACM, p. 9.

[197] Xie, H., Lui, J. C., Towsley, D. Incentive and reputation mechanisms for online crowdsourcing systems. In *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)* (2015), IEEE, p. 207–212.

[198] Yan, T., Marzilli, M., Holmes, R., Ganesan, D., Corner, M. mCrowd: A Platform for Mobile Crowdsourcing. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems* (New York, NY, USA, 2009), SenSys '09, ACM, p. 347–348.

[199] Yang, D., Xue, G., Fang, X., Tang, J. Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In *Proceedings of the 18th annual international conference on Mobile computing and networking* (2012), ACM, p. 173–184.

[200] Yang, K., Zhang, K., Ren, J., Shen, X. Security and privacy in mobile crowdsourcing networks: challenges and opportunities. *IEEE Communications Magazine 53*, 8 (2015), 75–81.

[201] Yu, B., Singh, M. P. Detecting deception in reputation management. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems* (2003), ACM, p. 73–80.

[202] Yu, H., Kaminsky, M., Gibbons, P. B., Flaxman, A. Sybilguard: defending against sybil attacks via social networks. *ACM SIGCOMM Computer Communication Review 36*, 4 (2006), 267–278.

[203] Yu, H., Kaminsky, M., Gibbons, P. B., Flaxman, A. D. Sybilguard: defending against sybil attacks via social networks. *IEEE/ACM Transactions on networking 16*, 3 (2008), 576–589.

[204] Yu, H., Miao, C., Shen, Z., Leung, C. Quality and budget aware task allocation for spatial crowdsourcing. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (2015), International Foundation for Autonomous Agents and Multiagent Systems, p. 1689–1690.

[205] Yu, H., Shen, Z., Miao, C., An, B. A reputation-aware decision-making approach for improving the efficiency of crowdsourcing systems. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (2013), International Foundation for Autonomous Agents and Multiagent Systems, p. 1315–1316.

[206] Zacharia, G. *Collaborative reputation mechanisms for online communities*. Tese de Doutorado, Massachusetts Institute of Technology, 1999.

[207] Zhang, X., Yang, Z., Sun, W., Liu, Y., Tang, S., Xing, K., Mao, X. Incentives for mobile crowd sensing: A survey. *IEEE Communications Surveys & Tutorials 18*, 1 (2015), 54–67.

[208] Zhang, X., Yang, Z., Zhou, Z., Cai, H., Chen, L., Li, X. Free market of crowdsourcing: Incentive mechanism design for mobile sensing. *IEEE transactions on parallel and distributed systems 25*, 12 (2014), 3190–3200.

[209] Zhang, Y., van der Schaar, M. Reputation-based incentive protocols in crowdsourcing applications. In *INFOCOM, 2012 Proceedings IEEE* (2012), IEEE, p. 2140–2148.

[210] Zhao, D., Li, X.-Y., Ma, H. How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications* (2014), IEEE, p. 1213–1221.