UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DIEGO DA SILVA PEREIRA

UMA ARQUITETURA DE CLOUD ROBOTIC BASEADA EM CLONES PARA UMA EQUIPE DE CELLBOTS

NATAL - RN
2016

DIEGO DA SILVA PEREIRA

UMA ARQUITETURA DE CLOUD ROBOTIC BASEADA EM CLONES PARA UMA EQUIPE DE CELLBOTS

Orientador: Profº ANDERSON ABNER DE SANTANA SOUZA, D.Sc.

NATAL - RN
2016

P436a Pereira, Diego da Silva.
Uma Arquitetura de Cloud Robotic Baseada em Clones para uma Equipe de CellBots / Diego da Silva Pereira. - 2016.
64 f. : il.

Orientador: Anderson Abner de Santana.
Dissertação (Mestrado) - Universidade Federal Rural do Semi-árido, Programa de Pós-graduação em Ciência da Computação, 2016.

1. Cell Bot. 2. Ad Hoc. 3. Robótica na Nuvem. I. Santana, Anderson Abner de, orient. II. Título.

DIEGO DA SILVA PEREIRA

UMA ARQUITETURA DE CLOUD ROBOTICS BASEADA EM CLONES PARA UMA EQUIPE DE CELLBOTS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: 29 / 07 / 2016

Anderson Abner de S. Sousa

Prof. Dr. Anderson Abner de Santana Sousa
Orientador e Presidente

Prof. Dr. Aquiles Medeiros Filgueira Burlamaqui
Examinador Externo - UFRN

Profa. Dra. Rosiery da Silva Maia
Examinadora Interna - UERN

Dedico este trabalho aos meus pais, Antonia Salete da Silva Pereira e Francisco Nilson Pereira, e meus irmãos, Diogo da Silva Pereira e Daniela da Silva Pereira.

"...form ever follows function. This is the law."

Louis H. Sullivan (Arquiteto Americano)

AGRADECIMENTOS

Agradeço a Deus por me conceder saúde para concluir esta dissertação.

Agradeço ao meu orientador Prof. Dsc. Anderson Abner de Santana Souza por creditar confiança em mim e nesta pesquisa.

Agradeço a todos os professores do Programa de Pós-graduação UERN-UFERSA, em especial ao grupo da unidade de Natal, nas pessoas do Prof. Dsc. Wilfredo Blanco Figuerola, Profa. Dsc. Rosiery da Silva Maia e Profa. Dsc. Cláudia Maria Fernandes Araújo Ribeiro pelo esforço e empenho no desenvolvimento do PPgCC dentro da capital do estado.

Agradeço a Universidade do Estado do Rio Grande do Norte(UERN) e ao Laboratório de Aprendizagem Robótica(LAR) por ceder a infraestrutura física para esta pesquisa. Estendo os agradecimentos a todos aqueles que compõe o laboratório, em especial Bruno Agenor Soares Santana.

Sou grato ao Institudo Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN) - Campus Parnamirim por conceder o afastamento para capacitação pelo tempo necessário para concluir esta dissertação.

Agradeço a todos que de forma direta ou indireta colaboraram para este trabalho tornar-se realidade.

RESUMO

Este trabalho retrata a implementação de uma arquitetura de comunicação *Ad Hoc* para uma equipe de CellBots como infraestrutura para utilização de computação na nuvem. CellBots são robôs que fazem uso de *smarthphones* como computador controlador aliados a dispositivos eletrônicos simples e de baixo custo. O objetivo é expandir os horizontes de aplicações para estes dispositvos robóticos através da formação de um MRS (*Multi-Robot System*) permitindo que eles possam trabalhar de forma cooperativa na conclusão das mais diversas tarefas. Além disso, é empregado o modelo de CR (*Cloud Robotic*) baseado em Clones para facultar a utilização de recursos oriundos da computação na nuvem através de robôs clone instanciados sob demanda, permitindo aumentar a capacidade global de processamento do MRS, viabilizar o compartilhamento de conhecimento, garantir uma maior automia para as baterias dos robôs através da redução dos gastos com o processamento local de informações, dentre outros benefícios. Na comunicação dentro do MRS foi adotado o protocolo AODV (*Ad Hoc On-Demand Distance Vector*) e para prover os clones na nuvem foi escolhida a plataforma *Rapyuta*. Para validação foram feitos testes de RTT (*Round Trip Time*) em variados cenários. A arquitetura sagrou-se viável tanto para comunicação dentro do MRS quanto para comunicação dos robôs com a nuvem.

Palavras-chave: CellBots, *Ad Hoc*, Robótica na Nuvem.

ABSTRACT

This work about the implementation of an ad hoc communication architecture for a Cellbots team as infrastructure to allow the use of cloud computing. Cellbots are robots that have a financial low cost because they use smarthphones as a central control unit and require few electronic components for their construction. The goal is to expand the horizons of applications for these devices through an architecture that enables communication between robots, allowing them to work as a team, cooperatively, like a Multi-Robot System (MRS), performing different tasks, such as searching objects. Another objective of the proposed architecture is to provide the Cellbots the use of resources from cloud computing, treated in the literature as Cloud Robotics.The expected results is to increase the overall capacity of system processing, to enable knowledge sharing and increase automia battery of robots by reduction spending on local processing information. It is proposed a clone based approach in order to achieve the benefits cited with the Cloud Robotics tecnology. The Rapyuta framework was used to instanciete clones and setting up their parameters. To provide communication between robots of the team was adopted the AODV(Ad Hoc On-Demand Distance Vector) protocol. At the end manuscrit some experiments are show to validade the proposed architecture.

Key-words: Cellbots, Cloud Robotic, Ad Hoc.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A robótica é resultado da busca constante por benefícios para aumentar a qualidade de vida humana e, em poucas décadas de atividade, ela já se faz presente em diversos setores da sociedade. Seja no setor industrial, de saúde, de segurança, ou ainda, de entretenimento, ela sempre busca promover melhorias e avanços. Tal motivação, fez com que esta área seja foco de diversos centros de pesquisa e laboratórios de tecnologia, os quais visam a construção de máquinas-robôs que possam auxiliar ou substituir os humanos nas mais variadas tarefas. Thrun, Burgard e Fox (2005) definem a Robótica como sendo a ciência de perceber e manipular o mundo físico através de dispositivos controlados computacionalmente, os robôs. Khalil e Dombre (2004) descrevem uma entidade robótica como um sistema mecânico, reprogramável, controlado de forma automática com diversos graus de liberdade, que pode ser fixo em um local ou móvel.

Na década de 60, onde apareceram as primeiras aplicações robóticas, o foco era otimizar a execução de tarefas repetitivas, tediosas ou até mesmo perigosas para o homem. Por exemplo, tarefas nas linhas de montagem de indústrias automobilísticas, como pinturas de chassis e soldagem. Os robôs executavam uma sequência de tarefas de forma repetitiva em alta velocidade e com boa precisão, para tal, tinham formato e comportamento semelhante ao braço humano, sendo, assim, classificados como robôs manipuladores (ROCHA, 2006). Porém, já trabalhava-se com entidades robóticas capazes de se mover dentro de um ambiente, são os chamados robôs móveis. Um dos primeiros exemplos de aplicação envolvendo robôs móveis foi automatizar o transporte de materiais dentro de ambientes industriais através da utilização de veículos de condução automática (ARTHUR, 1966).

Além da capacidade de locomoção, diversos recursos foram integrados a estes robôs, como poder computacional, comunicação sem fio, uma grande variedade de sensores, entre outros. Em consequência disso, diferentes aplicações foram atribuídas aos robôs móveis, tais como vigilância de ambientes, sejam internos ou externos (urbanos) (LIU; WANG; FENG, 2005), manutenção e descontaminação de indústrias nucleares (SAVALL; AVELLO; BRIONES, 1999), exploração de locais inóspitos (BELLINGHAM; RAJAN, 2007), busca e salvamento de pessoas e animais (MURPHY, 2004), aplicações militares (MARCHANT et al., 2011), etc.

A capacidade de locomoção dos robôs logo foi ampliada, permitindo que estas entidades fossem inseridas nos mais diversos ambientes. Com isso, a classe de robôs móveis foi subdivida em terrestres, aéreos e aquáticos. Então, um novo conjunto de aplicações são possíveis de serem executas, como pesquisa e exploração oceânica (WHITCOMB, 2000), cartografia e topografia de grandes áreas florestais (KOH; WICH,

2012), monitoramento de área de incêndios (AMBROSIA et al., 2003), entre outras. A Figura 1 apresenta três exemplos de avanços na área da robótica que permitiram avanços consideráveis para a sociedade, sendo: a Figura 1(a) a utilização de robôs teleoperados para realização de procedimentos cirúrgicos de forma remota, permitindo que paciente e médico esteja a kilômetros de distância; a Figura 1(b) a presença da robótica nas grandes linhas de produção industrial, viabilizando um maior desempenho nas linhas de montagem de insdústrias automobilísticas e, ainda, a retirada do ser humando de postos de trabalhos com alto grau de periculosidade; por fim, a Figura 1(C) retrata o emprego dos robôs na exploração de locais inóspitos e de difícil acesso, nesse caso, em missões espaciais ao planeta Marte.

Figura 1 – Exemplos da utilização da robótica para promover qualidade de vida: (a) na área da saúde; (b) no setor industrial; (c) em pesquisas espaciais.

Fonte: Próprio Autor.

Normalmente, independente da aplicação que o dispositivo robótico é empregado, o modo de operação de um robô é baseado na utilização de seus sensores para coletar informações do ambiente no qual está inserido (sistema de percepção), seguido pelo processamento dessas informações com objetivo de obter um resultado (planejamento), caso necessário, faz uso do seu sistema de locomoção para mover-se ou seus atuadores (garra, mão mecânica, braço manipulador, etc) para agir sobre o ambiente (ROCHA, 2006). Tal modo de operação é ilustrado de forma genérica através de um fluxograma na apresentado na Figura 2.

Figura 2 – Esquema genérico da hierarquia do modo de operação de um robô.

Fonte: Próprio Autor.

Entretanto, existem situações nas quais a utilização de um único robô compromete a execução ou a completude de uma missão. Então, com intuito de ampliar o alcance

funcional destes robôs e, visando melhorar a eficiência na execução das tarefas, começou-se a trabalhar com a possibilidade de agir com um grupo de entidades robóticas atuando juntas, de forma organizada, e com um único objetivo. É nesse contexto que surge a Robótica em Rede, tratada na literatura como *Networked Robotics*. Conforme Hu, Tay e Wen (2012), um sistema de Robótica em Rede é um grupo de dispositivos robóticos conectados através de uma rede de comunicação de dados, seja ela cabeada ou sem fio. Tais sistemas são classificados em dois grupos, Teleoperados e Multirrobôs (*Multi-Robots Systems* – MRS). No primeiro caso, os dispositivos que compõem o sistema são operados remotamente por humanos, como por exemplo, robôs utilizados em procedimentos cirúrgicos na área da saúde descritos por Taylor et al. (1995). No segundo caso, uma equipe de robôs autônomos executa uma tarefa de forma cooperativa, onde cada membro envia dados aos demais robôs do sistema via uma tecnologia de redes. Tal estrutura, por exemplo, pode ser empregada para acelerar o processo de mapeamento de uma determinada área, como feito por Michael et al. (2012) no mapeamento de prédios com infraestrutura comprometida por terremotos no Japão.

MRS possuem vantagens quando comparados aos sistemas com um único robô, como: 1) maior distribuição espacial e decomposição de problemas: diversos robôs ocupando segmentos espaciais de uma grande área, onde cada um possui recursos próprios (poder computacional, sensores, capacidade de percepção, locomoção e atuadores) necessários para realização de uma subtarefa, que quando concluída fará parte do resultado final obtido pelo sistema; 2) robustez e custo monetário: determinadas missões apresentam alto grau de complexidade e exigem uma quantidade considerável de recursos distintos, tais como diferentes tipos de sensores e mobilidade terrestre e aquática. Portanto um MRS permite uma alta flexibilidade na equipe e, com isso, obtém uma distribuição do risco, viabilizando capacidades heterogêneas e, até mesmo, redundância entre os robôs, tornando o sistema mais robusto e tolerante a falhas. Além disso, como há uma tendência que cada robô da equipe seja mais simples, afinal não é necessário que cada um deles seja equipado com todas as capacidades exigidas pela missão, o custo para o MRS pode ser reduzido (ROCHA, 2006).

Logo, uma missão complexa pode ser atribuída a uma equipe de robôs inteligentes e autônomos que a decomponham em tarefas mais simples e, através de cooperação, alcançam o êxito na sua conclusão. Assim, pode-se verificar os MRS sendo aplicados em várias tarefas como evidenciam os trabalhos de Rocha (2006), Fenwick, Newman e Leonard (2002) e Fox et al. (2006).

Conforme exposto, embora os sistemas cooperativos multirrobôs apresentem as vantagens mencionadas, eles são bastante desafiadores, pois, além de herdar todos os problemas associados ao desenvolvimento de um único robô, eles levantam novos problemas de pesquisa. Tais como (ROCHA, 2006):

1. percepção cooperativa através da integração de observações dos diferentes robôs;

2. arquitetura da equipe (por exemplo, centralizado ou controle distribuído) e como alcançar coerência e desempenho no controle e a coordenação do sistema, uma vez que os dados são normalmente distribuídos e cada robô tem apenas uma visão parcial do mundo;

3. planejamento cooperativo pela decomposição de tarefas complexas e atribuição das subtarefas para os robôs individuais;

4. assegurar um comportamento coerente na equipe quando ocorrem eventos inesperados (por exemplo, falhas de robôs, perturbações ambientais, etc.).

Então, prover comunicação entre entidades robóticas móveis independentes, com intuito de permitir que elas atuem de forma cooperativa não é trivial. Nesse sentido, é necessário fornecer mecanismos que possibilitem que tais dificuldades sejam superadas afim de garantir o melhor desempenho possível para o sistema.

Apesar das grandes vantagens apresentadas pelos MRS, esses robôs autônomos enfrentam algumas limitações físicas inerentes ao processamento de dados realizado a bordo do robô, visto sua capacidade computacional e, ainda, o acesso restrito aos dados via rede para tornar viável o armazenamento coletivo do sistema. Uma proposta para solucionar parte dessas limitações é oriunda do forte avanço nas tecnologias de redes sem fio, aliado às recentes inovações em tecnologias de computação em nuvem, trata-se da Robótica na Nuvem (*Cloud Robotics* - CR). Dessa maneira, pretende-se reduzir o custo computacional para cada robô da rede, aumentar a disponibilidade dos dados e, consequentemente, proporcionar uma maior autonomia para os robôs. Inquestionavelmente, *Cloud Robotics* é um tema de pesquisa fascinante, porém é necessário identificar suas aplicações alvo e encontrar seus domínios particulares, tais como *Big Data* (MANYIKA et al., 2011), *Cloud Computing* (MELL; GRANCE, 2012), Aprendizagem Coletiva (PANAIT; LUKE, 2005), entre outras. Acredita-se que CR é uma forma de impulsionar a cognição no domínio da robótica (CLARK; GRUSH, 1999), o que é fortemente apoiado pela União Europeia através de diversas chamadas de artigos de pesquisas (JORDAN et al., 2013).

Uma arquitetura de *Cloud Robotics* permite superar as limitações do MRS com recursos elásticos oferecidos por uma infraestrutura de nuvem onipresente. Isso significa que através da computação em nuvem é possível estender os recursos utilizados pela robótica em rede, tais como poder de processamento e capacidade de armazenamento conforme a necessidade do robô. O *National Institute of Standards and Technology* (NIST)(MELL; GRANCE, 2012) define computação em nuvem como:

> um modelo para permitir, conveniente, acesso ubíquo sob demanda da rede para um *pool* compartilhado de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com um mínimo de esforço de gerenciamento ou interação com o prestador de serviços.

A partir de seus três modelos de serviço: SaaS (*Software as a Service*), PaaS (*Platform as a Service*) e IaaS (*Infrastructure as Service*), a computação em nuvem permite uma enorme flexibilidade na concepção e implementação de novas aplicações para a robótica em rede (HU; TAY; WEN, 2012).

Com isso, pretende-se transferir tarefas computacionalmente intensivas para a execução na nuvem; unificar um grande volume de informações oriundas dos robôs para obter uma visão global do ambiente que eles estão inseridos; e, ainda, fornecer uma extensa biblioteca de habilidades e comportamentos que estão relacionadas com exigências da tarefa, tornando viável aprender com a experiência de todos os robôs ativos no MRS. Exemplos de arquiteturas são apresentadas e avaliadas por Hu, Tay e Wen (2012) e Osunmakinde e Ramharuk (2014).

Há um importante fator que dificulta o desenvolvimento de tecnologias e pesquisas relacionadas à robótica, mais ainda aquelas que envolvam sistemas multirrôbos. Trata-se da aquisição ou da construção dos componentes do sistema, os robôs, principalmente quando pretende-se trabalhar com um grupo deles. Logo, ao pensar na construção desses sistemas, deve-se levar em conta a quantidade de equipamentos necessários, que será multiplicada pelo número de robôs a serem desenvolvidos. Uma interessante alternativa é o uso de celulares como computador controlador dos robôs aliados a dispositivos eletrônicos simples e de baixo custo. Isso possibilita que os robôs tomem ao seu favor todos os sensores e características dos celulares modernos, os quais são frequentemente equipados com *Global Positioning System* (GPS), câmeras, acelerômetros, bússolas, *Wi-Fi*, *bluetooth*, microfone, altofalantes, entre outras características que os habilita a figurar como um sistema de controle para os robôs (AROCA et al., 2012).

Já foi comprovada a capacidade desses equipamentos (*smartphones*) executarem complexos algoritmos de navegação robótica (SANTOS; TARRATACA; CARDOSO, 2010). Os robôs baseados nessa abordagem são chamados de *CellBots* e, segundo Guizzo e Deyle (2012), os robôs baseados em *smartphones* correspondem a uma das principais tendências da robótica.

A grande problemática para prover *Cloud Robotics* para *Multi-Robot System* está vinculada, principalmente, a três pré-réquisitos:

1. Robôs: o primeiro pré-requisito refere-se aos robôs. Quais entidades robóticas são mais adequadas para o contexto o qual a pesquisa está inserida? Existe uma grande limitação orçamentária? Os robôs serão construídos ou será adotada uma plataforma robótica pronta? Que sensores e atuadores estes dispositivos devem possuir embarcados? São questionamentos razoáveis que um pesquisador deve levar em consideração antes de iniciar qualquer tipo de projeto nessa área;

2. Comunicação entre os robôs: após falar da unidade básica do MRS, é fundamental preocupar-se como prover a comunicação entre os elementos que compõe o

sistema multirrobôs. Qual melhor interface de comunicação adequa-se com as características do robô utilizado? Como será formado esse MRS? A comunicação é de curta ou longa distância? Que dados irão transitar pelo canal de comunicação? Quais protocolos serão adotados para permitir o trâmite desses dados pelo enlace de comunicação? São itens extremamente importantes para garantir um funcionamento adequado de um MRS e de qualquer tipo de comunicação interna ou externa a ele;

3. Utilização de *cloud computing*: por fim, deve-se garantir a utilização de algum modelo de *cloud computing* para otimizar o desempenho do MRS. É fundamental perceber como garantir a utilização dos benefícios da computação na nuvem sem inserir limitações ao funcionamento do MRS, logo é preciso montar uma infraestrutura capaz de atender as demandas necessárias para garantir a troca de dados entre a nuvem e os robôs sem comprometer o funcionamento do sistema;

Diante dos itens expostos anteriormente, está pesquisa propôs uma arquitetura de comunicação que teve como entidade robôtica adotada os *CellBots*, os quais utilizam para comunicação interna e externa ao MRS a interface de comunicação IEEE 802.11n e, além disso, fazem uso da plataforma robótica de *Cloud Robotics* chamada *Rapyuta*, a qual foi implementada a partir do modelo de *cloud computing* baseado em PaaS (*Platform as a Service*). Os detalhes da solução são apresentados nos capítulos seguintes.

1.1 CONTRIBUIÇÕES

Neste trabalho é apresentada uma arquitetura de *Cloud Robotics* para um sistema multirrobôs composto por *CellBots* homogênios, fornecendo-lhes a autonomia para execução da tarefa localmente, utilizando seus próprios recursos internos, ou remotamente, com utilização dos recursos de computação na nuvem.

O objetivo inicial é permitir que a equipe de *CellBots* seja capaz de realizar tarefas de busca, considerando que o mapa do ambiente está previamente carregado na nuvem junto com a missão. Além de proporcionar aos robôs a troca de mensagens dentro do MRS com intuito de realizar cooperação, afim de concluir a tarefa da forma mais eficiente.

Nesse sentido, é feita uma breve análise de qual a melhor interface de comunicação sem fio deve ser adotada para utilização em tais robôs, e também qual arquitetura apresenta resultados mais satisfatórios para permitir comunicação dentro da equipe de robôs e com a infraestrutura de nuvem.

Para a comunicação interna do MRS foi adotado o protocolo AODV (*Ad Hoc On-Demand Distance Vector*), visto sua capacidade de adaptação e por consumir uma quantidade pequena de recursos dos nós que compõe a rede quando comparado com outros protocolos empregados em MANET (*Mobile Ad Hoc Networks*). Além de ser responsável em manter a infraestrutura de rede do MRS funcionando, o AODV encapsula informações adicionais que são utilizadas pelos *CellBots* para realização de tarefas cooperativas. Para tal foi inserido um campo multidimensional em seu cabeçalho com informações sobre os *status* do robô (disponível ou ocupado) e da tarefa (em execução, concluída e abortada).

Na comunicação externa ao MRS, realizada via *websocket* entre cada membro da equipe de *CellBots* e a nuvem, foi usada a plataforma *Rapyuta* (MOHANARAJAH et al., 2015a). Com isso, foi possível transferir grande parte do custo computacional para execução de tarefas, como reconhecimento de objetos, para uma clone virtual do robô criado na infraestrutura de computação na nuvem.

1.2 ESTRUTURA DO TRABALHO

Esta dissertação está organizada em 6 seções. Sendo a primeira seção destinada à Introdução, a qual apresenta uma contextualização do tema pesquisado, bem como uma breve abordagem da arquitetura de comunicação utilizada para permitir os *CellBots* utilizarem recursos da computação na nuvem. O Capítulo 2 contempla a fundamentação teórica sobre comunicação *ad hoc* em redes de *CellBots*, destacando a estrutura dos robôs utilizados no MRS, as formas de comunicação *wireless* que podem ser empregadas, principais conceitos que envolvem sistemas multirrobôs e suas formas de funcionamento, e, por fim, um detalhamento de computação na nuvem. O Capítulo 3 retrata os trabalhos relacionados com o tema, dando ênfase às arquiteturas já em uso. O Capítulo 4 destaca a arquitetura de comunicação utilizada, dando destaque à sua organização e ao seu funcionamento. O Capítulo 5 aborda os cenários de testes e resultados. E finalmente, o Capítulo 6 contém as considerações finais e expectativas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Paredis et al. (2002) destaca que para ocorrer colaboração é necessário coordenar e coletar informações para que membros individuais de uma equipe de robôs móveis possam agir como uma única entidade lógica. Por isso, é fundamental que o sistema robótico seja organizado e capaz de adaptar-se para permitir a comunicação eficiente entre os seus nós (membros). Para Cao, Fukunaga e Kahng (1997), dada uma tarefa, um sistema multirrobôs exibe comportamento cooperativo se, devido a algum mecanismo subjacente, ou seja, mecanismo que permite a cooperação, há um aumento na utilidade total do sistema, por exemplo, execução da tarefa em menor tempo. Logo, caso tal mecanismo não fosse empregado, o ganho não seria possível, pois este influência na dinâmica de interação entre os agentes do sistema.

Diante disso, é necessário adotar estratégias de comunicação entre os robôs do MRS e dos robôs com a infraestrutura de nuvem, visto que elas devem operar de forma adequada para viabilizar o funcionamento coeso do sistema de modo a não comprometer a eficiência na obtenção do resultado final da missão. Para tal, é necessário levar em consideração uma diversidade de itens que estão diretamente relacionados com este processo, tais como canal de comunicação, interfaces de rede disponíveis, topologias mais eficientes, serviços que serão ofertados, entre outros. Portanto, este Capítulo aborda a relação desses itens com as entidades robóticas utilizadas na implementação do sistema multirrobôs, os *CellBots*. É dado destaque às tecnolgias sem fio viáveis para comunicação entre cada *CellBot* e os demais componentes do sistema, sejam eles outros *CellBots*, atraveś da comunicação R2R (*Robot-to-Robot*), ou a nuvem, via comunicação R2C (*Robot-to-Cloud*), e também as principais arquiteturas adotadas para prover *Cloud Robotics*.

2.1 UTILIZAÇÃO DE CELLBOT

CellBot é um robô que possui como unidade central um *smartphone* e faz uso dos recursos do aparelho para realizar suas missões. Devido a isso, possui uma arquitetura de *hardware* simples, porém com plataformas computacionais poderosas, permite que os pesquisadores possam se concentrar em suas pesquisas e testes, em vez de se preocuparem com a montagem e integração de sensores, atuadores e infraestrutura de conexão (AROCA et al., 2013). A Figura 3 apresenta um *CellBot* desenvolvido no Laboratório de Aprendizagem Robótica (LAR) da Universidade do Estado do Rio Grande do Norte (UERN), Campus Natal.

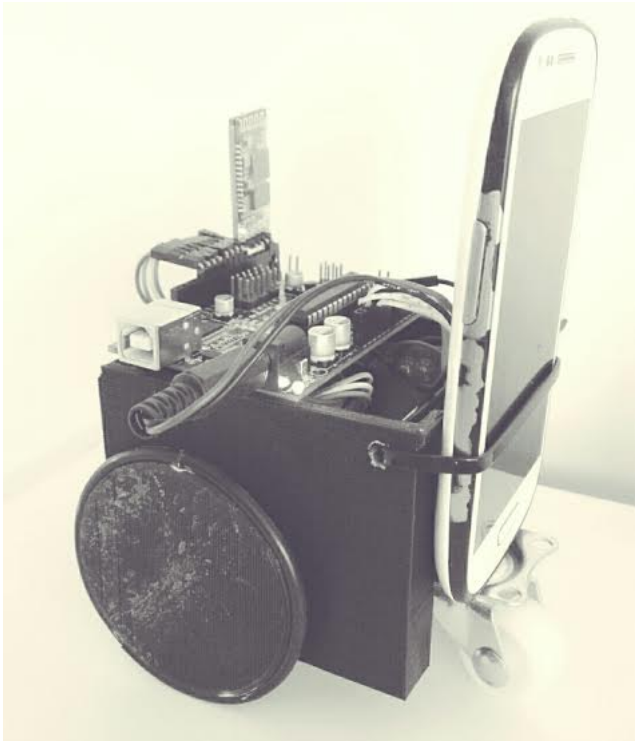

Figura 3 – CellBot desenvolvido no Laboratório de Aprendizagem Robótica (LAR) na UERN (Natal/Brasil).

Fonte: Próprio Autor.

Além dos itens já apresentados, os *CellBots* têm configurações atrativas para desenvolvimento das mais diversas aplicações robóticas, como destaque tem-se o reconhecimento de voz e a visão computacional. Santos, Tarrataca e Cardoso (2010) demonstram que é viável a utilização de algoritmos complexos para navegação robótica em tais dispositivos. Isto é possível pois a grande maioria dos *smartphones* atuais são equipados, no mínimo, com processadores *multicores* de 1 GHz e 1 GB de memória RAM.

Outro fator que torna viável o desenvolvimento de aplicações para essas área é a presença do Android. O Android é um sistema operacional de código aberto baseado em Linux e desenvolvido pela Google para *smartphones* e *tablets*. A fim de proporcionar uma pluralidade de aplicativos, é possível desenvolve-los usando Java e facilmente testa-los e implanta-los em dispositivos. O desenvolvimento pode ser feito com um simulador ou com dispositivos reais. Se necessário, o programador pode usar outras linguagens, tais como C, *Python* ou qualquer outra suportada pelo Linux (AROCA et al., 2012).

Outra motivação para trabalhar com dispositivos baseados no Android é sua integração à maioria das aplicações na nuvem. Espera-se que aplicativos estejam disponíveis para capacitar estruturas mecatrônicas com um toque no botão, graças a *Cloud Robotics*. Com isso, os *smartphones* podem realmente atuar como o "cérebro"do

robô (JORDAN et al., 2013).

Uma API (*Application Programming Interface*) Android oferece diversas possibilidades para os desenvolvedores, por exemplo, reconhecimento de voz, uma característica extremamente desejada para robôs. A partir da API é possível sintetizar a voz no idioma desejado e em frases complexas com apenas algumas linhas de código. Além disso, o acesso a câmeras e sensores de orientação é bastante simplificado. Existem funções para avaliar a presença dos recursos do robô, visto que *smartphones* apresentam diversidades em sua configuração. Em 2011, a Google lançou o Android *Open Accessory Development Kit*, que permite o desenvolvimento de acessórios externos para interagir com os dispositivos Android, inclusive celulares (AROCA et al., 2012).

Os *CellBots*, conforme descrito anteriormente, possuem como unidade controladora um *smarthphone*. As interfaces de comunicação presentes nestes dispositivos são, normalmente, infravermelho, *bluetooth* e *Wi-Fi* (IEEE 802.11). Portanto, obrigatoriamente, os robôs precisam utilizar uma dessas tecnologias para comunicar-se com os membros da equipe e com a nuvem, quando necessário. Sendo assim, é preciso verificar qual delas é mais viável para adoção, conforme os recursos presentes nos robôs e as tarefas que estes serão submetidos.

2.2 TECNOLOGIAS DE COMUNICAÇÃO SEM FIO

Dentre os principais fatores que estimularam o rápido crescimento e popularização das tecnologias de redes sem fio, destacam-se a flexibilidade e a mobilidade na rede. De acordo com Fernando, Loke e Rahayu (2013), os principais protocolos de conexão para comunicação em *Cloud Robotics* são *Wi-Fi*, *bluetooth* e 3G (Rede de Terceira Geração para Telecomunicações). Com isso, esta seção apresenta detalhes destas tecnologias de comunicação *wireless*, e, por fim, uma comparação apresentando justificativas para o emprego de uma delas neste trabalho.

2.2.1 Tecnologia de Comunicação Sem Fio Wi-Fi

O IEEE (*Institute of Eletrical and Eletronics Engineers*) possui um comitê responsável por tratar de especificações técnicas para redes locais (*Local Area Network* - LAN) e metropolitanas (*Metropolitan Area Network* - MAN), o 802. O IEEE 802.11 descreve apenas especificações para redes local sem fio (*Wireless* LAN - WLAN), porém ficou popularmente conhecida como *Wi-Fi* (*Wireless Fidelity*), graças à marca registrada *Wi-Fi*

Alliance, fundada em 1999 pelas empresas Aironet, 3Com, Lucent Tecnologies, Nokia e Symbol Tecnologies. Ela está dividida conforme algumas características, como largura de banda, frequência e desempenho. Além disso, existem modificações que foram empregas com intuito de promover melhoria no processo de comunicação sem fio que ficaram agrupadas em gerações, sendo organizadas com o seguinte padrão: 802.11x, onde x irá identificar a tecnologia específica que está em operação, por exemplo, 802.11b. As gerações e suas especificações técnicas, como largura de banda, frequência e modulação estão descritas em Crow et al. (1997).

Atualmente, a interoperabilidade entre os padrões IEEE 802.11a/b/g/n domina o mercado de redes WLAN. A tecnologia apresenta melhores resultados em ambientes internos sem obstáculos, curtas distâncias, e com um número restrito de assinantes. Fernandes et al. (2012) apresenta detalhes das condições ideais de utilização e uma avaliação interessante sobre o comportamento da tecnologia nos mais variados cenários.

A tecnologia *Wi-Fi* possui duas arquiteturas básicas: a infraestruturada, ilustrada Figura 4, com a presença de um elemento central (base) e a *Ad Hoc*, retratada na Figura 5, sem a presença deste elemento. A estratégia empregada pelo IEEE 802.11 é permitir mobilidade de forma transparente para as camadas superiores (SCHILLER, 2003).

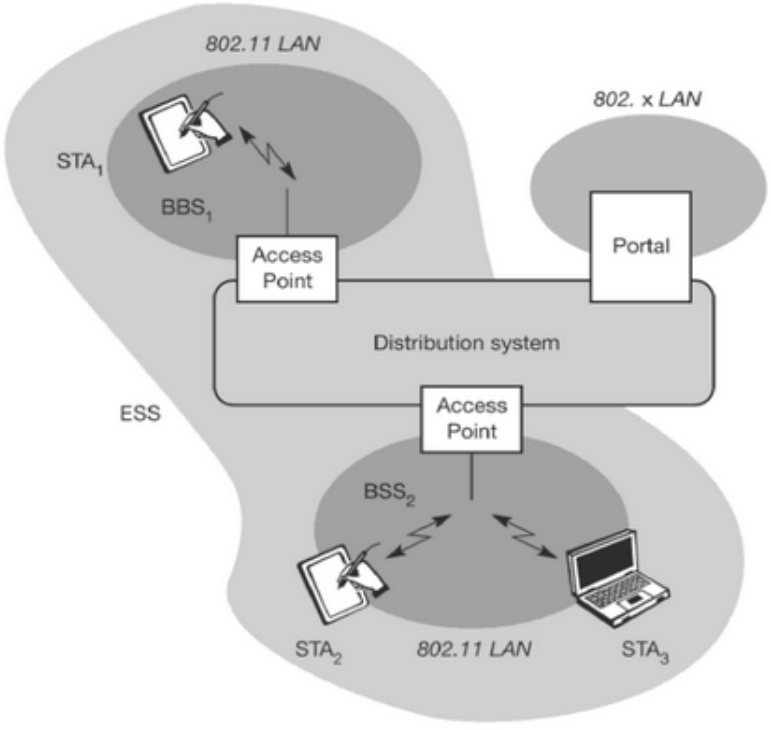

Figura 4 – Rede Sem Fio Infraestruturada e Seus Componentes.

Fonte: (SCHILLER, 2003).

Uma rede *wireless* infraestruturada é formada pelos seguintes componentes: BSS (*Basic Service Set*), STA (*Station*), AP (*Acess Point*) e ESS (*Extended Service Set*). A estação

móvel (STA) e o Ponto de Acesso (AP) são os principais componentes descritos pela IEEE 802.11, onde a STA é o dispositivo que, normalmente, é referido como um adaptador de rede ou cartão de *interface* de rede (*Network Interface Card* – NIC) e liga a rádio frequência ou meio sem fio. O AP é responsável em viabilizar a conexão do meio sem fio com a rede cabeada, bem como a forma de credenciamento para tal acesso (SCHILLER, 2003).

Todas as redes 802.11 estruturadas são construídas em torno das BSS, ou seja, conjunto básico de serviços. Trata-se de um grupo de STA tentando conectar-se entre si, no entanto é necessário utilizar o SSID (*Service Set Identifier*) para identificar o BSS que deseja-se ingressar. O ESS, como o próprio nome sugere, é um nível acima do BSS na arquitetura 802.11. É possível ter diversas STA, formando múltiplos BSS, e AP com intuito de gerar uma WLAN maior, formando um conjunto de serviços estendidos, o ESS (SCHILLER, 2003).

As redes *Ad Hoc* surgiram com intuito de atender a demandas militares oriundas do DARPA (*Defense Advanced Research Projects Agency*) no EUA, que pretendia prover uma estrutura de comunicação entre elementos móveis sem a necessidade de um dispositivo central, coordenando o acesso ao canal de comunicação pelos demais elementos que compõe a rede. Para tanto, cada nó da rede deve ter a capacidade de atuar como um roteador, executando funções referentes a descoberta e escolha de rotas para diferentes destinos, além de participar do repasse de dados em trânsito pela rede (SCHILLER, 2003).

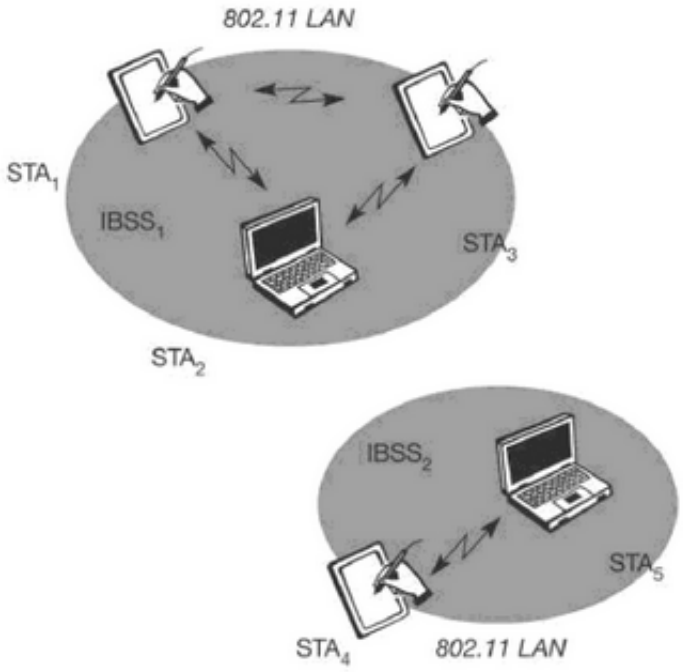

Figura 5 – Rede Sem Fio *Ad Hoc*.

Fonte: (SCHILLER, 2003).

Diferente das redes estruturadas, as redes *Ad Hoc* são formadas por agrupamentos

de equipamentos organizados em IBSS (*Independent Basic Service Set*) com função semelhante dos BSS das redes estruturadas, permitindo comunicação direta apenas com os elementos do mesmo IBSS. No entanto, caso haja necessidade de comunicar-se com outros IBSS é preciso que algum elemento da rede seja considerado *gateway* e tenha uma outra *interface* de rede pertencente ao IBSS alvo (HAAS et al., 2002).

2.2.2 Tecnologia de Comunicação Sem Fio Bluetooth

Conforme Schiller (2003), o *Bluetooh* é um padrão global de comunicação sem fios de curto alcance que permite conectividade entre umas vasta quantidade de dispositivos eletrônicos, e vem sendo empregado nas mais diversas área, como na computação, através dos computadores, *tablets* e seus acessórios, na telefonia, com os *smarthphones* e telefonia móvel, tecnologia médica, automação residencial, aplicações no âmbito esportivo, etc.

Segue padrão IEEE 802.15.1 e foi projetado com dispositvos de baixo custo. Está diretamente relacionado com aplicações em redes WPAN (*Wireless Personal Area Network*), conhecida como rede de área pessoal. Duas topologias de conectividade são definidas para o padrão: a *piconet* e *scatternet*. Uma *piconet* é uma WPAN formada por um dispositivo *bluetooth* mestre (*master*) e um ou mais dispositivos escravos (*slaves*). Todos os dispositivos participantes nas comunicações de uma determinada *piconet* são sincronizados com o *clock* do mestre. Os escravos comunicam-se em ponto-a-ponto apenas com o mestre e mediante autorização dele. Já as transmissões do mestre podem ser ponto-a-ponto ou ponto-para-multiponto. Outra característica da tecnologia é a possibilidade de mudança de estado do nó, pois quando não está envolvido em um processo de comunicação (modo ativo), o nó *bluetooth* fica em modo de espera (estacionado), reduzindo, com isso, o consumo de energia.

A *scatternet* é um conjunto de *piconets* operacionais sobrepostas no tempo e no espaço. Duas *piconets* podem ser ligados para formar uma *scatternet* e um dispositivo pode participar de várias *piconets* ao mesmo tempo, permitindo, assim, a possibilidade de que a informação possa fluir para fora da área de cobertura de uma *piconet*. Um dispositivo em uma *scatternet* pode ser um escravo em várias *piconets*, mas mestre em apenas uma delas. Um cenário típico de organização de redes que utilizam a tecnologia *bluetooth* é apresentado na Figura 6 (LEE; SU; SHEN, 2007).

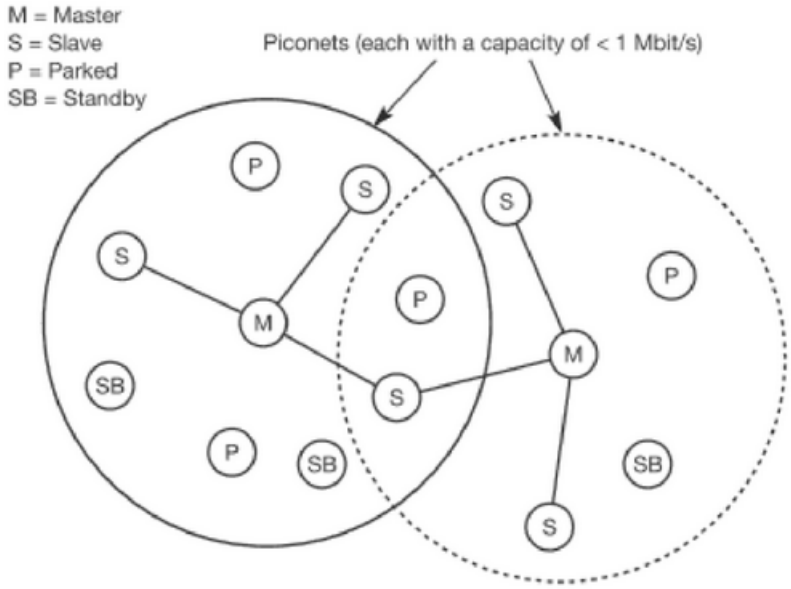

Figura 6 – Rede Sem Fio *Bluetooth* organizada em *Piconets* e *Scatternet*.

Fonte: (SCHILLER, 2003).

2.2.3 Comparação entre Wi-Fi e Bluetooh

O *Wi-Fi* e *Bluetooth* operam na banda não licenciada de 2,4 GHz ISM. O *Wi-Fi* foi inicialmente projetado para substituição do cabeamento e para compartilhamento de periféricos (como impressoras, dispositivos de armazenamento compartilhado) entre computadores e demais dispositivos em redes locais sem fio. O *Bluetooth*, por outro lado, foi destinado para equipamentos e aplicações não-residentes, como fones de ouvido, mouses, teclados na rede de área pessoal.

Nesse cenário, *o Wi-Fi* tem maior alcance, com um raio de 100m e suporta até 11 Mbps taxas de dados. O *Bluetooth* é caracterizada por exigir baixa potência e um baixo custo do *chips transceptor*, porém seu raio de cobertura é normalmente de 10 m, dependendo da classe do dispositivo e da presença de obstáculos físicos no ambiente. Por fim, o 3G é uma tecnologia para provedores de serviços móveis voltada para telecomunicações, e, inicialmente, não apresenta características satisfatórias para atender um MRS.

Logo, entre as opções expostas e recursos presentes nos *CellBots*, esta pesquisa será desenvolvida utilizando como padrão de comunicação sem fio as especificações descritas por IEEE 802.11. A Tabela 1 demonstra um comparativo entre esses protocolos de conexão.

Protocolo	Vantagens	Desvantagens
Wi-Fi	Superior alcance de comunicação Baixo consumo de energia	Preocupações em relação à segurança e questões de interoperabilidade entre as diferentes marcas *Wi-Fi*
Bluetooth	Menor uso de energia e maior disponibilidade	Alcance de comunicação limitado
3G	Cobertura quase universal	Alto consumo de energia e aumento da latência, resultando em um desempenho insatisfatório

Tabela 1 – Comparação entre Protocolos de Conexão usados em *Cloud Robotics*.
Fonte: (OSUNMAKINDE; RAMHARUK, 2014)

2.3 PROTOCOLOS DE ROTEAMENTO PARA MANET

2.3.1 Mobile Ad Hoc Network (MANET)

MANET (*Mobile Ad Hoc Network*) consiste em plataformas móveis equipadas com *interfaces* de comunicação sem fio, designadas simplesmente como nós, que são livres para se movimentar de forma arbitrária e capazes de trocar informações entre si através de uma rede *ad hoc*. É um sistema autônomo de nós móveis que pode operar de forma isolada, ou ter acesso para uma rede cabeada (CORSON; MACKER, 1998).

Para Corson e Macker (1998), as principais características pertencentes a uma MANET são:

- Topologia dinâmica: os nós são livres para se moverem de forma arbitrária; assim, a topologia da rede, que é tipicamente *multihop*, pode alterar aleatoriamente e rapidamente, por vezes imprevisível.
- Limitação de largura de banda: canais de comunicação sem fios têm capacidade de transmissões significativamente menores do que a cabeada. Além disso, a taxa de transferência, após os efeitos de acesso múltiplo, desvanecimento, ruído e demais condições de interferência, é menor do que a taxa máxima de transmissão.
- Restrições de energia: alguns ou todos os nós em uma MANET podem fazer uso de baterias ou outras fontes limitadas de energia. Com isso, um dos critérios mais importantes de projeto do sistema deve ser otimização para conservação de energia.
- Limitação de segurança: redes móveis sem fio são geralmente mais propensas a ameaças à segurança física do que as redes por cabo fixo. O aumento da

possibilidade de ataques de espionagem, falsificação e de negação de serviço deve ser cuidadosamente considerado. Como benefício, a natureza descentralizada de controle da rede proporciona robustez contra pontos únicos de falha.

Normalmente, as redes *ad hoc* móveis são aplicadas em lugares onde há pouca ou nenhuma infraestrutura de comunicação, ou ainda, a infraestrutura existente é cara ou de utilização inconveniente. O conjunto de aplicações para MANET é diversa, variando em escala, forma de mobilidade, tipos de dispositivos que compõe a rede, formas de fontes de energia, etc. Goyal, Parmar e Rishi (2011) afirmam que uma grande variedade de aplicações é possível, como por exemplo: viabilizar comunicação entre diversos soldados no campo de batalha; coordenar de equipes de salvamento em situações de desastre; criar um rede multimídia instantânea e temporária para compartilhamento de dados entre dispositivos diversos, como *smartphomes* e *notebooks*; interconectar eletrodomésticos em uma residência; aplicações robóticas para criação de equipes de robôs por demanda, entre outras.

2.3.2 Roteamento em MANET

Protocolos de roteamento são algoritmos compostos por rotinas responsáveis em mapear a topologia da rede. Normalmente, o mapeamento é realizado atraveś da utilização de uma tabela de roteamento, a qual sua forma de construção, atualização e manutenção varia conforme o método de roteamento adotado. Diversos itens devem ser considerados ao tratar-se de roteamento em MANET, destacando-se a possibilidade de mobilidade dos nós com a consequente falta de informações prévias a respeito da localização deles e, ainda, a não obrigatoriedade de que todos os nós estejam na mesma área de cobertura. Ou seja, é possível que alguns elementos da rede não tenham conhecimento de todos os nós que compõe a rede que estão associados. Logo, essas redes são previstas para ter um comportamente altamente dinâmico (HEIDEMANN et al., 2001).

Esses protocolos estão divididos em três grupos (ABOLHASAN; WYSOCKI; DUTKIEWICZ, 2004):

1. Pró-ativos: também conhecidos como *Table Driven* ou orientados a tabela, os protocolos desse grupo têm como a principal característica a descoberta e manutenção de rotas através do envio periódico de pacotes. Ou seja, um intervalo de tempo constante irá indicar o momento para enviar pacotes de consulta sobre o estado da rede. Com isso, cada nó sempre tem conhecimento global da rede que ele está inserido. Isso permite um menor atraso no envio do primeiro pacote, porém requer

muito mais recursos do nó, são menos escaláveis e causam maior *overhead* na rede devido à grande quantidade de informações de controle. São exemplos desses protocolos o *Destination Sequence Distance Vector* (DSDV) (HE, 2002) e o *Optimized Link State Routing Protocol* (OLSR) (CLAUSEN; JACQUET, 2003).

2. Reativos: tratados por *Source-Initiead On-Demand Driven*, iniciados sob demanda, este grupo tem como critério principal a busca por rota somente quando é necessário realizar entregas de dados a um determinado destino. Com isso, o processo de busca de uma rota válida é iniciado conforme a demanda de entrega para determinado nó. Portanto, tem um maior tempo de entrega para o primeiro pacote, devido ao tempo de descoberta da rota. No entanto, possui larga escalabilidade, além de consumir menos recursos do nó e consumir menos banda da rede com informações de controle. Seus principais representantes são o *Ad Hoc On-Demand Distance Vector* (AODV) (PERKINS; BELDING-ROYER; DAS, 2003) e o *Dynamic Source Routing Protocol* (DSR) (JOHNSON et al., 2001).

3. Híbridos: protocolos híbridos fazem uso das características dos grupos anteriores, pró-ativos e reativos, para seu funcionamento. Um exemplo é o *Zone Routing Protocol* (ZRP) (HAAS; PEARLMAN; SAMAR, 2002).

Em uma grande quantidade de trabalhos, tais como *Proactive or reactive routing: a unified analytical framework in MANETs* (WU et al., 2008), *Performance study of ad hoc routing protocols with gossip-based approach* (LEE; RA; KIM, 2009) e *Comparative performance analysis of DSDV, AODV and DSR routing protocols in MANET using ns2* (TUTEJA; GUJRAL; THALIA, 2010), foram feitas comparações nos mais variados formatos entre os procolos citados anteriormente, sendo constatado que o protocolos reativos apresentam melhor desempenho, principalmente quando refere-se a escalabilidade, utilização de energia e ocupação de largura de banda da rede, do que os protocolos pró-ativos. Além disso, destaca-se ainda que o protocolo AODV apresentou melhor desempenho que o DSR, portanto é adotado nesta viabilizando a entrega de dados na rede de robôs móveis.

2.3.3 Protocolo Ad Hoc On-Demand Distance Vector (AODV)

O protocolo AODV tem características peculiares que o torna viável para esta pesquisa. Dentre elas, destacam-se: caráter adaptativo, que permite uma rápida adaptação diante de mudanças na topologia da rede; simplicidade, devido à quantidade reduzida de mensagens utilizadas em seu funcionamento; volatilidade, por manter informações de rota apenas quando o nó precisa ou faz uso dela. Tudo isso faz com que ocorra uma economia considerável dos recusos dos nós que compõem a rede.

Por tratar-se de um protocolo reativo, sempre que se deseja enviar pacotes a um nó destino que não consta em sua tabela de roteamento, inicia-se o processo de descoberta de rotas. Para ilustrar esse processo de forma detalhada, a Figura 7 mostra as principais mensagens envolvidas, representadas pelas setas, bem como as tabelas de roteamento dos nós envolvidos, sendo, os nós representados pelos círculos nomeados sequenciamente da A até D, e as tabelas pelos retângulos, que são atualizadas a partir da transmissão das mensagens AODV na rede. Para facilitar o entendimento, foi utilizado um cenário com apenas quatro nós. Para o sucesso do mecanismo é necessário a execução dos seguintes passos:

1. O nó "A" precisa enviar pacotes ao nó "D";
2. "A" consulta sua tabela e verifica que não há rota para "D", logo o processo de descoberta de rota é iniciado;
3. O nó "A" envia um pacote *Route Request* (RREQ) em *broadcast* para todos os seus vizinhos;
4. As mensagens de RREQ propagam-se na rede até que o nó destino ou um nó intermediário que conheça a rota desejada por "A" seja encontrado;
5. O nó destino ou um nó intermediário que conhece a rota, responde à requisição com uma mensagem *Route Reply* (RREP) de volta ao nó de origem, no exemplo, "A"; Nesse processo, as tabelas de roteamento dos demais nós também vão sendo atualizadas com a informação da nova rota.
6. A mensagem RREP chega ao nó "A" que finalmente atualiza sua tabela e pode enviar pacotes ao nó "D".

Outro ponto importante do protocolo AODV é a manutenção das rotas contidas na tabela de roteamento através de um processo de validação. Como os nós são móveis, frenquentemente ocorrem modificações na topologia da rede, logo, uma rota traçada em algum instante de tempo pode perder validade em poucos instantes. Para tal, o AODV faz uso de envio periódico de mensagens *HELLO* entre os nós para verificar a existência ou ruptura de rotas. Quando um desses pacotes é aguardado por algum nó, porém não é recebido, é detectado um erro e imediatamente um pacote *Route Error* (RERR) é enviado ao nó de origem e para os demais nós que forem necessário notificar o erro. Com isso, todos os nós que receberem esta mensagem tomaram conhecimento que a rota não é mais válida (PERKINS; BELDING-ROYER; DAS, 2003).

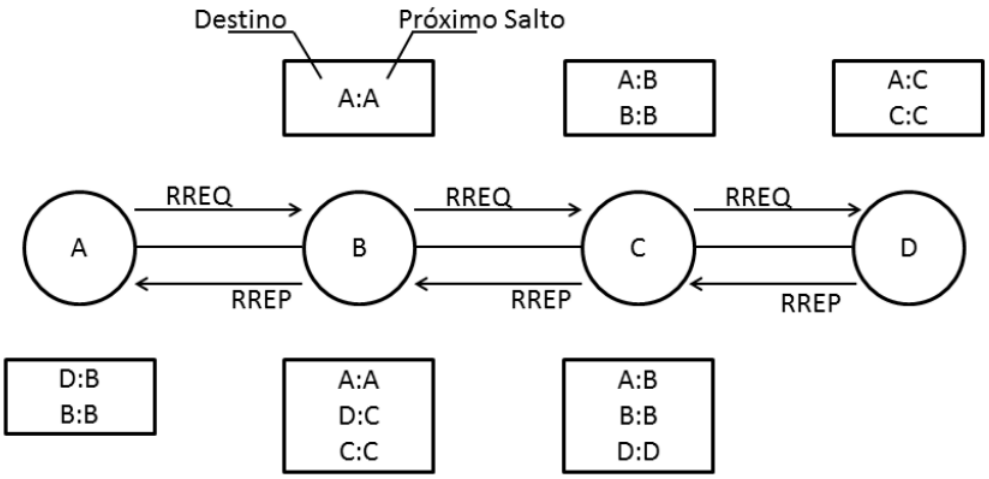

Figura 7 – Funcionamento do protocolo AODV durante o processo de descoberta de rota em uma rede com quatro nós.

Fonte: (JÚNIOR, 2014).

2.4 ARQUITETURAS PARA REDE ROBÓTICA E A NUVEM

2.4.1 Rede Robótica Ad Hoc

Rede Robótica trata, especialmente, de como os robôs presentes na rede irão detectar e distribuir a carga de trabalho necessária para conclusão de uma missão. Em consequência disso, é fundamental levar em consideração a forma que os robôs irão: ser acionados, comunicar-se com seus vizinhos e propagar informações após conclusão ou não da tarefa atribuída. Conforme Vandenberghe, Moerman e Demeester (2012), um Sistema de Robótica em Rede é um conjunto de recursos alocados em locais identificados de uma rede e, por meio de configuração apropriada, executam uma tarefa de forma cooperativa.

A partir do desenvolvimento dos dispositivos robóticos e tecnologias de comunicação sem fio, conseguiu-se flexibilizar e otimizar o funcionamento destes sistemas, inclusive, apropriando-se de vários conceitos e técnicas da área de comunicação *wireless* para permitir novas organizações das equipes de robôs. Vários padrões como *Zigbee* (ALLIANCE et al., 2006), *Bluetooth* e IEEE 802.11 (*Wi-Fi*) foram desenvolvidos para comunicações sem fio de curto alcance e, rapidamente, aplicados na robótica. Uma rede de robôs é muitas vezes formada dinamicamente e de forma *Ad Hoc*. Tal arquitetura

caracteriza-se pela ausência de um controlador central para coordenar o fluxo de comunicação na rede, onde os nós podem entrar e sair, ou ainda, tornar-se indisponíveis devido a falhas imprevisíveis ou obstruções no ambiente. Além disso, a rede é altamente dinâmica se os robôs são móveis, caso no qual enquadra-se esta pesquisa (HU; TAY; WEN, 2012).

Em alguns estudos publicados, os robôs são equipados com tecnologia IEEE 802.11 em combinação com um protocolo MANET existente. Esse protocolo permite que os robôs formem uma rede de modo *Ad Hoc*, ou seja, não há roteadores fixos para encaminhar mensagens. Logo, todos os nós são capazes de rotear mensagens e podem ser ligados de forma dinâmica e arbitrária. Nós dessas redes funcionam como roteadores que descobrem e mantêm rotas para outros nós na rede (VANDENBERGHE; MOERMAN; DEMEESTER, 2012).

No entanto, é necessário aprimorar os protocolos MANET e adaptá-los à realidade dos robôs, que em sua grande maioria não dispõe de muitos recursos, principalmente de automia energética (carga da bateria). Neste sentido, muitas pesquisas estão sendo desenvolvidas visando prover protocolos específicos para redes de robôs *Ad Hoc*, e é a partir disso que surge na literatura a Rede *Ad Hoc* Robótica (*Robotic Ad Hoc Network* - RANET).

Uma RANET deve ser capaz de operar sem qualquer infraestrutura fixa, permitindo que os robôs se encontrem automaticamente e configurem todas as tabelas de roteamento para garantir a comunicação necessária entre os membros da equipe. Para tal, é necessário adotar métricas de desempenho, normalmente incluem taxa de transferência, a probabilidade de perda de pacotes, atraso fim a fim, número médio de saltos, otimização dos saltos, sobrecarga de roteamento. Porém, o consumo de energia deve ser considerado como uma importante métrica de desempenho, que pode incluir informações na utilização de um caminho e verificar o tempo de vida energética dos nós robôs (KOUVATSOS; MISKEEN, 2012).

2.4.2 Arquiteturas para Robótica Nuvem

Todas as arquiteturas aqui apresentadas estão organizadas em dois níveis, os quais já foram mencionados anteriormente, o R2R (*Robot-to-Robotic*), que trata-se de um grupo de robôs comunicando-se via uma tecnologia *wireless* formando uma rede *ad hoc* colaborativa, e o R2C (*Robot-to-Cloud*), que é a infraestrutura física de nuvem, que fornece um conjunto de recursos computacionais alocados sob demanda (HU; TAY; WEN, 2012). Logo, as arquiteturas serão a combinação entre esses níveis, e suas peculiaridades irão abordar exatamente as formas distintas de agrupar as entidades nos

níveis e como esses níveis comunicam-se.

a) Modelo Baseado em Pontos (*Peer-Based Model*): cada robô e cada máquina virtual (VM) na nuvem é considerado como uma unidade de computação. A principal vantagem deste modelo de robótica na nuvem é que os robôs e VMs formam uma rede totalmente distribuída, permitindo que determinadas tarefas robóticas sejam divididas em pequenas tarefas que podem ser executadas em um subconjunto da rede distribuída. No entanto, a desvantagem do modelo é a descentralização da rede criada pelo mapeamento de um-para-um de robô e VM na nuvem, o que o torna difícil de administrar. Outra desvantagem é o baixo número de conexões de rede entre os robôs individuais e a infraestrutura de nuvem física (ver Figura 8);

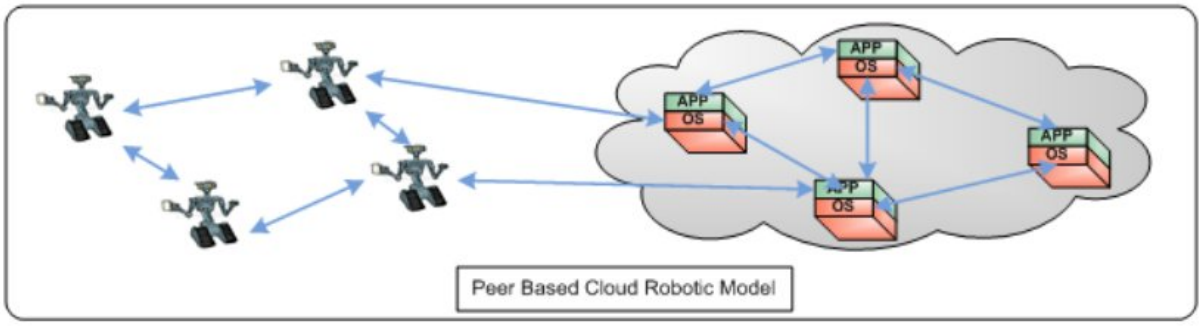

Figura 8 – Modelo de Robótica na Nuvem *Peer-Based*.

Fonte: (OSUNMAKINDE; RAMHARUK, 2014)

b) Modelo Baseado em Proxy (*Proxy-Based Model*): neste modelo, uma unidade da equipe multirrobôs funciona como um líder do grupo, sendo responsável pela comunicação com um *proxy* VM na infraestrutura de nuvem, gerenciando toda interação entre a rede de robôs e a nuvem. A principal vantagem deste modelo robótico baseado em *proxy* é a sua maior interoperabilidade em comparação com os outros modelos de robótica na nuvem. Interoperabilidade neste sentido, refere-se às complexidades adicionais causadas na manutenção de uma infraestrutura de robótica na nuvem. A principal desvantagem é o número de conexões entre os robôs e a nuvem. Neste modelo apenas uma conexão de rede pode ser estabelecida entre os dois níveis da arquitetura, logo, nos casos em que o líder do grupo não está disponível, a comunicação para a nuvem não será possível, comprometendo todo o sistema (ver Figura 9);

c) Modelo Baseado em Clone (*Clone-Based Model*): o modelo baseado em clone requer que cada robô do MRS tenha um clone correspondente no nível de infraestrutura física de nuvem. A principal vantagem deste modelo é que uma tarefa particular pode ser executada no clone na nuvem ou localmente no robô. A principal desvantagem do modelo é o aumento da complexidade provocada pela manutenção e operação da infraestrutura nuvem robótica, que reduz a sua interoperabilidade. Outra desvantagem são as migrações VM necessárias, a fim de aumentar a mobilidade de um robô (ver Figura 10);

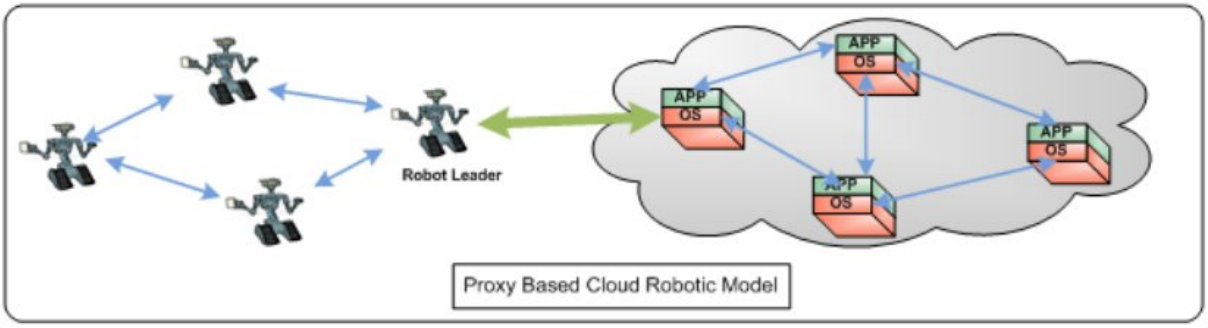

Figura 9 – Modelo de Robótica na Nuvem *Proxy-Based*.

Fonte: (OSUNMAKINDE; RAMHARUK, 2014)

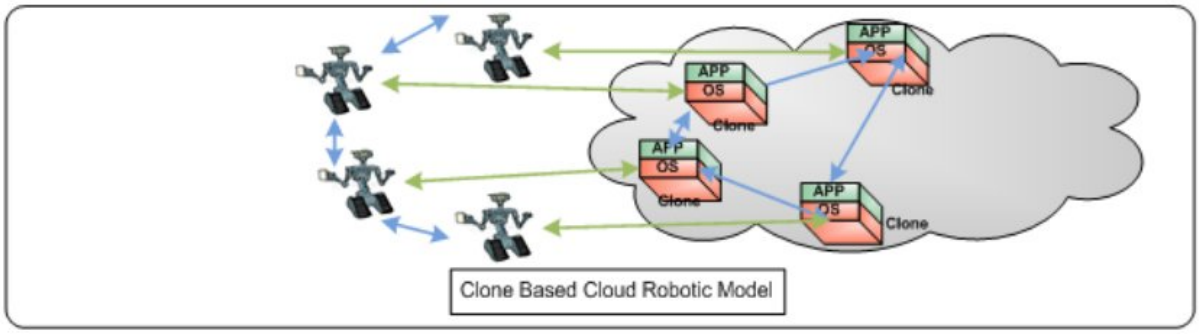

Figura 10 – Modelo de Robótica na Nuvem *Clone-Based*.

Fonte: (OSUNMAKINDE; RAMHARUK, 2014)

d) *Survivable Cloud Framework Multi-Robotics* (SCMR): este modelo trata-se de uma proposta para superar os problemas de falta de conectividade com a nuvem. Semelhante ao modelo baseado em *proxy*, ele apresenta um líder que assume as funcionalidade da infraestrutura física de nuvem em caso de desconexão, o que exige um poder computacional maior para este robô. No entanto, as demais unidades do sistema comunicam-se com seus clones na nuvem, normalmente, exigindo intervenção do líder caso a nuvem fique indisponível. Sua principal vantagem refere-se a apresentação de uma solução para os problemas de desconexão da equipe de robôs com a nuvem nuvem. Como desvantagem aponta-se a necessidade de um robô líder com um maior poder computacional, inibindo sua aplicação em MRS homogênios, e ainda, complexidade no sincronismo entre a nuvem e o líder.

Alguns itens são levados em consideração para classificar um modelo de robótica na nuvem, são eles: robustez, interoperabilidade e flexibilidade. a) Robustez: refere-se à quantidade de conexões entre o nível R2R e a nuvem. Logo, o modelo baseado em clone é o mais robusto, já que apresenta a maior quantidade de conexões. Por outro lado, o modelo baseado em *proxy* a menor, sendo o menos robusto. b) Interoperabilidade: trata-se da complexidade necessária para operações entre uma infraestrutura de nuvem e uma rede robótica. Com isso, o modelo baseado em *proxy* apresenta maior interoperabilidade devido a sua estrutura hierárquica, enquanto o baseado em clones apresenta a menor

menor. c) Flexibilidade: é a capacidade para suportar a mobilidade dos robôs. O modelo baseado em pontos (*peers*) possui mais flexibilidade devido à facilidade em instanciar VMs em qualquer lugar da infraestrutura de nuvem. Já o modelo baseado em clones é o menos flexível, visto que mecanismos de migração de VMs complicados são necessários para a mobilidade das unidades da equipe robótica.

O modelo SCRM apresenta característica dos modelos baseados em pontos e clones, porém, devido à necessidade de uma entidade robótica diferenciada para servir como clone da nuvem, não é levada em consideração para este trabalho, já que o MRS composto por *CellBots* são homogêneos, ou seja, possuem capacidades e funções semelhantes.

De acordo com Hu, Tay e Wen (2012), escolher um modelo específico de computação elástica depende principalmente de três fatores: condições da rede, requisitos da aplicação e disponibilidade de recursos. Este trabalho mostra uma estrutura unificada para determinar um modelo ideal, ou quase ideal, para um determinado conjunto de aplicações que envolvem *CellBots*. O escopo inicial é realizar uma busca dentro de uma área com mapa previamente conhecido e, posteriormente, reconhecer o objeto de busca.

2.4.3 Relação entre Cloud Computing e Cloud Robotics

Cloud Robotics é a fusão de tecnologias de computação na nuvem e robótica de serviços. A tecnologia de *Cloud Computing* (CC), ou simplesmente *Cloud*, é uma das áreas de maior crescimento das Tecnologias de Informação e Comunicação (TIC). Soluções baseadas em nuvem aparecem gradualmente em áreas de TIC, como o *Google Docs*, que permite acesso a *software* e armazenamento baseado em nuvem, um exemplo de *Software* como um Serviço (*Software as a Service - SaaS*). Da mesma forma, computação na nuvem também pode ser ofertada no formato de PLataforma como um Serviço (*Plataform as a Service* - PaaS) e Infraestrutura como um Serviço (*Infraestructure as a Service* - IaaS). Com base nestes termos, surgiu a Robótica como um Serviço (*Robotic as a Service* - RaaS), onde os sistemas robóticos estão envolvidos. O comum em todos esses modelos é ofertar dinamicamente um serviços para o usuário (JORDAN et al., 2013).

Aplicações SaaS são ofertadas através da internet, eliminando assim a necessidade de instalar e executar o aplicativo no sistema do usuário. São gerenciados a partir de um local centralizado e acessados remotamente por um navegador ou um cliente móvel. *Google Apps* é o conjunto de aplicativos mais utilizado via SaaS. PaaS refere-se a uma plataforma de computação ofertada por uma infraestrutura de nuvem. Permite aos desenvolvedores obter um conjunto de sistemas e ambientes necessários para o ciclo de vida de *software*, seja para desenvolver, testar, implantar e hospedar aplicações

web. Alguns exemplos são *Amazon Web Services* (AWS) e *Microsoft Azure*. IaaS fornece uma infraestrutura como um serviço. O cliente não precisa adquirir os servidores, *datacenters* ou os recursos de rede. A essência do modelo de IaaS é um modelo financeiro *pay-as-you-go*. *Amazon* e *Microsoft* também são fornecedores de IaaS. Estes modelos e a estrutura dos serviços estão apresentados na Figura 11 (KOKEN, 2015).

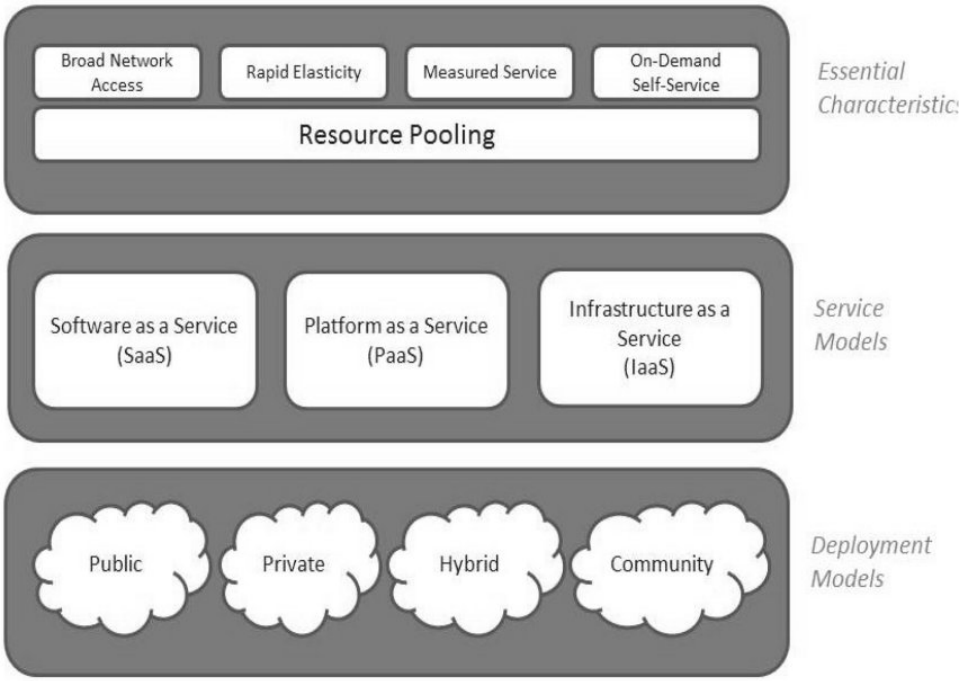

Figura 11 – Infraestrutura de Computação na Nuvem.

Fonte: (KOKEN, 2015)

Em virtude de toda essa disponibilidade de infraestrutura, surgiu outro conceito na robótica, os robôs *cloud-enabled*. Uma possível tradução para o português é "robôs habilitados pela nuvem". São entidades robóticas que utilizam em algum modelo de serviço a computação na nuvem, com isso, não ficam restritos aos recursos embarcados. Porém, este novo grupo de robôs, também classifica-se em duas categorias, robôs autônomos (*standalone*) e robôs em rede. A infraestrutura CR com robôs autônomos e robôs em rede é apresentada na Figura 12.

Robôs autônomos podem se beneficiar da nuvem em termos de poder de computação, capacidade de armazenamento e memória. No entanto, os robôs em rede, além das possibilidades já citadas para os autônomos, é possível ingressar/formar redes, compartilhar suas informações através da nuvem e realizar trabalhos colaborativos.

Tais inovações provocaram muita motivação na comunidade da robótica. É fácil observar isso em alguns discursos como Steve Cousins (CEO da *Willow Garage Incorporation*) "o papel da robótica na nuvem é fornecer uma oportunidade para uma comunicação aberta entre os robôs, já que nenhum robô é uma ilha". Ou ainda, Ken Goldberg (UC Berkeley) que enfatiza os benefícios da capacidade ilimitada de

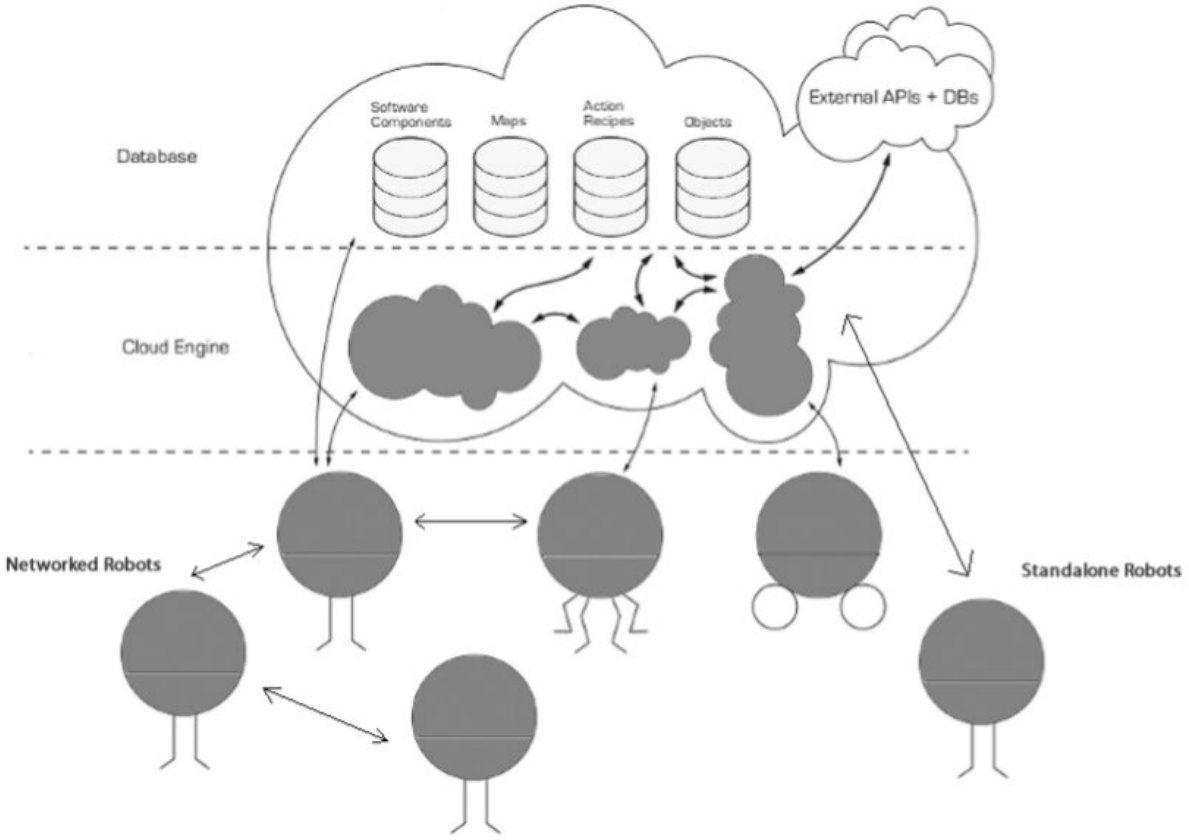

Figura 12 – Robôs Autônomos (*Standalone*) e Rede de Robôs utilizando recursos oriundos da computação na nuvem.

Fonte: (KOKEN, 2015)

computação, memória e programação, levando a uma nova forma de inteligência coletiva para robôs através da aprendizagem e partilha: "Os seres humanos como espécie estão ficando mais espertos porque somos capazes de compartilhar informações para construir inovações muito mais rápido. Os robôs têm esse potencial também". Mario Tremblay (CEO da *RobotShop*) afirma que "robótica na nuvem acontece quando nós conectamos robôs a internet e, em seguida, ao fazer isso, os robôs tem sua capacidade e inteligência aumentada. A nuvem lhes permite comunicar com outras máquinas e servir melhor os seus operadores humanos. Ao colaborar com outras máquinas e os seres humanos, os robôs transcendem suas limitações físicas e tornar-se mais úteis e capazes, uma vez que podem delegar parte de suas tarefas para situações mais adequadas "(JORDAN et al., 2013).

CR permite aos robôs beneficiarem-se do poder computacional, armazenamento, recursos de comunicação e de modernos *datacenters*. Também tornou possível que os robôs aproveitassem o rápido aumento das taxas de transferência de dados para descarregarem tarefas. O termo "*Cloud-Enabled Robotics*"foi apresentado por James Kuffner pela primeira vez no IEEE RAS Internation Conferência sobre *Humanoid Robotics* em 2010. Ele foi o primeiro a apontar o potencial de redes distribuídas combinadas com

a robótica, principalmente, para melhorar o robô. *Cloud Robotics* rapidamente ganhou impulso com iniciativas de empresas como Google, Willow Garage e Gostai, bem como mais de uma dúzia de projetos de pesquisa ativos em todo o mundo (MESTER, 2015).

Jordan et al. (2013) destacam cinco áreas onde o campo de *Cloud Robotics* deve avançar significativamente:

1. Robôs no mundo real precisam de uma grande quantidade de informações sobre o ambiente, pois possuem grande capacidade de exploração. O reconhecimento de vários objetos, sons, imagens, pessoas pode ser feito de forma muito mais eficiente com o auxílio da nuvem;

2. Navegação de robôs pode avançar muito, pois serviços como *Google Maps* são rápidos e confiáveis para que os robôs possam ultrapassar seus recursos internos. Serviços em nuvem podem revolucionar modelagem e planejamento de movimento, reduzindo custos e estabelecendo inovações;

3. A comunicação entre robôs também pode evoluir, tornando-se um componente crucial de um Sistema Multirrobôs. Os agentes podem partilhar sua informação sensorial, os resultados dos cálculos, notícias e suas experiências, adquiridas através de suas habilidades de raciocínio (podendo avançar para o nível cognitivo). Esta aplicação poderá ser semelhante a um "*Facebook* para robôs", onde os robôs podem compartilhar registros e sempre a melhor opção será escolhida pelo agente;

4. Precisa-se encorajar as pessoas a gerar códigos de fonte aberta, partilhar os seus planos, experiências e resultados. De acordo com os EUA *Robotics Roadmap*, um grande crescimento dessas comunidades é esperado nos próximos 5 anos.

5. O trabalho humano deve ser coordenado entre o desenvolvedor e o operador, pois em caso de erro, é mais fácil encontrar soluções.

Osunmakinde e Ramharuk (2014) destacam os principais obstáculos e desafios que precisam de ser superados referentes aos modelos de *Cloud Robotics*, pois serviços de robótica na nuvem são efetivamente limitados, ou seja, estes serviços são dependentes da largura de banda para o descarregamento de dados ou do número de conexões paralelas permitidos para a nuvem. Alguns pontos merecem destaque, como:

- Limitação do número de indivíduos em um MRS;
- Maximizar a eficácia dos recursos disponíveis na nuvem com o *on-demand*;
- Compatibilidade de recuperação de dados é um desafio constante, provocada por diferentes robôs que acessam a nuvem para executar em tarefas;

- Segurança e confiabilidade são outro desafio em curso na área da robótica na nuvem. O ambiente de servidor de nuvem tem de ser um ambiente com confiabilidade, não permitindo que as tarefas sejam interrompidas ou acessadas por dispositivos não autorizados.

2.4.4 Plataforma de Robótica na Nuvem Rapyuta

Rapyuta é baseado em um modelo de computação elástica, ou seja, aloca dinamicamente ambientes de computação, chamados clones, para robôs. Estes ambientes de computação estão fortemente interligados, permitindo a cada elemento do sistema multirrôbos compartilhar serviços e informações com os demais membros da equipe. Além disso, são compatíveis com o ROS, evitando dependências de *middlewares* personalizados e incompatibilidade entre os pacotes utilizados. A comunicação entre o robô e a plataforma é feita através de *websocket* (HUNZIKER et al., 2013).

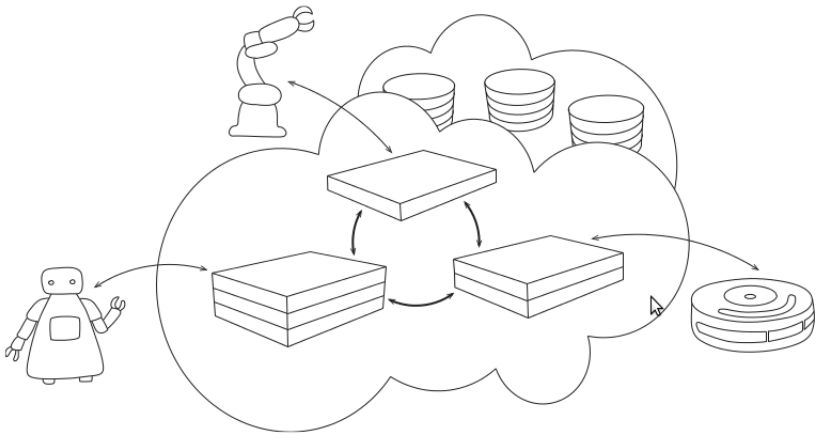

Figura 13 – Visão geral da Plataforma de *Cloud Robotic Rapyuta*.

Fonte: (HUNZIKER et al., 2013).

A Figura 13 apresenta uma visão simplificada da Plataforma *Rapyuta*, onde cada robô conectado à plataforma tem um ou mais ambientes de computação, representados pelos retângulos, facultando-lhes a possibilidade de transferir cargas computacionais para a nuvem. Esses ambientes de computação são interligado uns com os outros e podem fazer uso de repositórios de armazenamento também disponíveis na nuvem (discos circulares empilhados).

Existem três componentes principais na composição da *Rapyuta*: os ambientes de computação, um conjunto de protocolos de comunicação e tarefas do núcleo de

administração do sistema (MOHANARAJAH et al., 2015a). Tais componentes são brevemente descritos a seguir.

Ambientes de computação da *Rapyuta* são implementados usando *Linux Containers* (LXC), que fornecem uma solução leve e personalizável para a separação de processos, segurança e dimensionamento. O LXC isola os processos e recursos do sistema dentro de uma única máquina virtual. É configurado para executar qualquer processo em um nó ROS, permitindo a utilização do próprio mecanismo de comunicação desse *middleware* para interligar todo os nós presentes.

O conjunto de protocolos de comunicação procuram estabelecer um equilíbrio entre a complexidade do sistema e as latências resultantes da troca de dados entre os componentes. Faz uso de processos *Endpoint* que definem interfaces de forma clara para se comunicar com processos externos. No entanto, os *Endpoints* estão diretamente conectados uns com os outros utilizando portas de comunicação, o que otimiza consideravelmente o fluxo de dados.

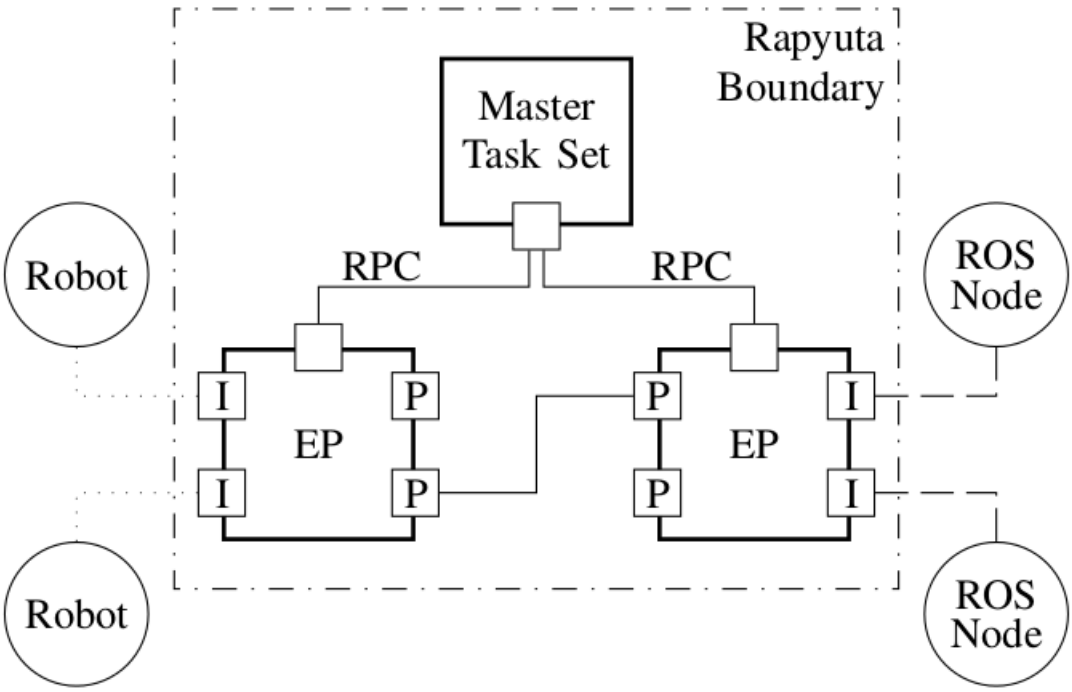

Figura 14 – Formas de Comunicação na Plataforma de *Cloud Robotics Rapyuta*.

Fonte: (MOHANARAJAH et al., 2015a).

Um *Task Set*, conjunto de tarefas, são funcionalidades do sistema. Uma ou mais *Task Set* podem ser colocadas para execução juntas como um único processo, dependendo do caso de uso. O *Master Task Set* é o processo que gerencia todo o sistema, sendo responsável por coordenar a conexão entre o robô e a plataforma *Rapyuta*, o estado da rede, processar requisições oriundas dos robôs, entre outras. O *Robot Task Set* é utilizado para prover as condições necessárias para manter a comunicação entre a

plataforma e o robô, criando a ponte de acesso entre o robô real e o seu clone na nuvem. O *Environment Task Set* foi definido para prover todas as capacidades necessárias para prover comunicação com o ambiente de computação, ou seja, comunicação entre os nós ROS e demais *EndPoints*. Por fim, o *Container Task Set* é utilizado para iniciar e finalizar os *containers* em execução na infraestrutura de nuvem.

A Figura 14 ilustra os canais de comunicação básicos da *Rapyuta*, onde os pontos finais (*EndPoints* - EP) estão conectados ao *Master Task Set* através de um protocolo de Chamada de Procedimento Remoto (*Remote Procedure Call* - RPC). Além disso, os terminais têm *Interfaces* (I) para conexões com robôs ou nós ROS e Portas (P) para a comunicação entre os EP. As linhas pontilhadas representam a comunicação externa via *websocket*, já as linhas tracejadas, por sua vez, representam a comunicação baseada em ROS entre nós ROS e a Rapyuta. Por fim, todas as linhas contínuas representam a comunicação interna entre os processos da plataforma.

Diversos trabalhos recentes fazem uso da Plataforma *Rapyuta* para prover *Robotics Cloud*. Destaca-se os trabalhos de Shinde, Borle e Longjam (2016), DePalma e Breazeal (2016) e Salmerón-García et al. (2016), fazem uso dos recurso da plataforma para otimizar aplicações que envolvem visão computacional para robôs móveis. Diante disso, comprova-se a viabilidade de utilizá-la para ofertar serviços de planejamento de caminho e reconhecimento de objetos para uma equipe de *CellBots*.

3 TRABALHOS RELACIONADOS

Diversos trabalhos exploram o uso de tecnologias de computação em nuvem para dar suporte a aplicações robóticas, porém, a diversidade entre elas está figurada na forma que a nuvem é integrada ao sistema multirrobôs. A grande dificuldade é adotar uma estratégia para descarregamento das tarefas na nuvem conforme a demanda da aplicação, pois vários fatores devem ser considerados, como quantidade de dados trocados, atrasos na propagação de mensagens, alocação de recursos na nuvem virtual formado pelo grupo de robôs, ou na infraestrutura física de nuvem composta por máquinas virtuais.

Hu, Tay e Wen (2012) ressaltam que o objetivo sempre é minimizar a quantidade de energia consumida pelo robô, sob a restrição de que a tarefa deve ser concluída dentro de um prazo especificado. O *trade-off* está entre a energia consumida pelo processamento local da tarefa e a energia consumida pela transmissão para a nuvem. A partir disso, é apresentado três arquiteturas possíveis para permitir a utilização de computação na nuvem por uma equipe de robôs, as quais baseiam-se na combinação de uma nuvem robótica *ad hoc* formada por comunicação *Robot-to-Robot* (R2R) entre os robôs que compõem o MRS, e uma infraestrutura de nuvem habilitada pela comunicação *Robot-to-Cloud* (R2C), composta de máquinas virtuais (*Virtual Machines - VMs*). Isto permite grande flexibilidade na concepção dos modelos, três deles são detacados: *Peer-Based*, *Proxy-Based* e *Clone-Based*, todos já detalhados no capítulo anterior.

Osunmakinde e Ramharuk (2014) apresentam o *framework* SCRM (*Survivable Cloud Multi-Robot*) que foi projetado para lidar com um dos grandes desafios presentes em arquiteturas de *Cloud Robotic*: a perda de conectividade com a infraestrutura de nuvem. Até o momento, as arquiteturas admitiam que a conexão entre o robô e a nuvem sempre estaria disponível, no entanto, quando múltiplos robôs atuam em ambientes heterogêneos nem sempre essa conexão pode ser garantida. Em consequência disso, o SCRM propõe a utilização de um robô líder que assume determinados serviços caso a nuvem fique *offline*. Este líder é tratado como um clone da infraestrutura da nuvem física, logo, ele irá preencher a lacuna entre a camada R2R e R2C por todo o período de indisponibilidade da nuvem. O SCMR usa como protocolo de comunicação *WebSockets* entre os robôs individuais e o servidor na nuvem.

Rapyuta (MOHANARAJAH et al., 2015a) é um *framework* de robótica na nuvem estruturado conforme o *Platform as a Service* (PaaS). Nesse formato, é possível terceirizar parte ou todos os processos computacionais a bordo do robô para um *data center*. A arquitetura deste *framework* é baseada em clones (*clone-based*) e suas principais diferenças para outros de estrutura semelhante, como o *Google App Engine*, são sua adaptação para multiprocessamento em alta largura de banda voltado para aplicações robóticas e a

possibilidade de ser modificado para suprir a demanda dos mais variados cenários, visto que é *open-source* e bem documentado. Além disso, por prover recursos sob demanda, é altamente escalável, proporcionando o comportamento elástico da nuvem, e também fornece ambiente para acessar o repositório do *RoboEarth*, o qual armazena e compartilha modelos de objetos, mapas de ambientes e ações para as mais variadas plataformas robóticas. Uma vez iniciado, os robôs podem se autenticar no *Rapyuta*, criar um ou mais ambientes computacionais garantidos na nuvem, e lançar os processos desejados.

Hunziker et al. (2013) descreve que os ambientes de computação criados na *Rapyuta* também podem ser construídos em arquiteturas de computação paralelas, caso as aplicações façam uso deste recurso. O protocolo de comunicação, baseado em *WebSockets*, prevê mecanismos de comunicação síncronas e assíncronas, e não são restritos a robôs baseados em ROS. *Browsers* e *smartphones* também podem conectar-se ao sistema. Ambientes de computação *Rapyuta* são privados, seguros e otimizados para transferência de dados. No entanto, sua performance é, em grande, parte determinado pela latência e qualidade da conexão de rede e o desempenho do *data center*. Destaca-se em aplicações de mapeamento colaborativo 2D/3D (MOHANARAJAH et al., 2015b), planejamento e execução de tarefas, reconhecimento de objetos, localização e teleoperação, entre outros.

Outro *framework* para *Cloud Robotics* é o Agentes Distribuídos com Inteligência Coletiva (*Distributed Agents with Collective Intelligence* - DavinCi) que foi desenvolvido buscando vantagens em relação à escalabilidade e ao paralelismo computacional para serviços robóticos em grandes ambientes (ARUMUGAM et al., 2010). Semelhante a *Rapyuta*, apresenta arquitetura baseada em Pontos (*Peer-Based*) e oferta serviços conforme *Service as a Software* (SaaS). Sua implementação em torno do *cluster Hadoop* (WHITE, 2012) com a arquitetura distribuída do ROS, permite aos agentes heterogêneos compartilhar dados de sensores e fazer o *upload* de dados para nuvem, visando processamento de algoritmos computacionais intensos. Nesse contexto, é possível explorar aspectos como paralelismo de algoritmos para mapeamento. A literatura já apresenta ganhos significativos no desempenho em tempos de execução para construção de mapas de grandes áreas utilizando o *FastSLAM* (MONTEMERLO et al., 2002). Com isso, o mapa global pode ser compartilhado com outros robôs através de SaaS, reduzindo a carga do sistema com a exploração e a necessidade de sensores adicionais para todos os robôs do MRS.

Para desenvolvimento da arquitetura aqui apresentada, foi necessário fundir características de parte dos trabalhos comentados anteriormente. O objetivo foi propor uma arquitetura que apresentasse meios de comunicação eficientes e adequados ao MRS formados por *CellBots*. É importante ressaltar que esses dispositivos robóticos possuem características construtivas que dependem, essencialmente, dos recursos dos *smartphones*. Assim, procurou-se otimizar a utilização dos recursos de *software* e *hardware* disponíveis nos *CellBots* para proporcionar ao grupo de robôs um ambiente colaborativo

no cumprimento de missões.

Dentre as principais mudanças, está a adoção de um protocolo de roteamento para redes móveis *ad hoc* no funcionamento e na comunicação dentro do sistema multirrobôs, além da utilização da plataforma *Rapyuta* com múltiplos clones para executar os algoritmos de reconhecimento de objetos e planejamento de caminho. Dessa forma, a entrega de mensagens dentro do MRS não utiliza o algoritmo *Gossip* (SHAH, 2009), como proposto por Hu, Tay e Wen (2012) e Osunmakinde e Ramharuk (2014), e sim o protocolo AODV (*Ad Hoc On-Demand Distance Vector*) (PERKINS; BELDING-ROYER; DAS, 2003). Apesar de não ser foco desta pesquisa, a adoção do AODV apresenta ganhos de performance em relação à latência, visto que o primeiro não utiliza tabelas de roteamento, baseando-se apenas em um fator probabilístico para entregar mensagens aos seus vizinhos.

No âmbito da plataforma de nuvem, foram desenvolvidos serviços, planejamento de caminho e de reconhecimento de objetos, que são ofertados para os robôs com intuito de deslocar o custo computacional para clones *Rapyuta*. Isso permite uma execução mais rápida dos algoritmos, devido ao maior poder de processamento na nuvem e, ainda, uma economia da carga das baterias dos *CellBots*.

4 ARQUITETURA BASEADA EM CLONES PARA UMA EQUIPE DE CELLBOTS

O bom funcionamento de um sistema que envolve uma equipe de robôs móveis e *Cloud Robotics* está diretamente relacionada à dois itens: o primeiro trata de como viabilizar a comunicação dentro da equipe de robôs móveis, já o segundo, da forma pela qual será realizada a conexão desses dispositivos com a nuvem. Nesse sentido, uma alternativa para comunicação entre os robôs é fazer uso dos recursos ofertados pelos protocolos MANET. Afinal, todos os *CellBots* utilizados possuem a tecnologia IEEE 802.11 embarcada e apresentam características semelhantes as *Wireless Sensor Network* (WSN) (YICK; MUKHERJEE; GHOSAL, 2008), tais como, limitação do tempo de vida da fonte de energia (bateria) e aleatoriedade no funcionamento da rede. Essas características são extramente importantes quando se lida com entidades robóticas móveis com recursos limitados, principalmente no que se refere à autonomia da bateria dos componentes do MRS.

A arquitetura aqui proposta tem como base os trabalhos de Hu, Tay e Wen (2012) e Osunmakinde e Ramharuk (2014), onde o sistema é composto por duas camadas, brevemente descritas anteriormente: R2R e R2C. A camada R2R é uma rede *ad hoc* colaborativa formada por *CellBots* homogêneos e não hierárquicos, ou seja, apresentam capacidades e funções semelhantes dentro do sistema e não há qualquer relação de hierarquia entre eles. A camada R2C, por sua vez, tem a função de ofertar os serviços e recursos computacionais para os robôs. Dentre os serviços ofertados estão o planejamento de trajetória, navegação e reconhecimento de objetos. Para disponibilizá-los aos *CellBots* foi adotada a plataforma *Rapyuta*, detalhada por Mohanarajah et al. (2015a) e Hunziker et al. (2013). Caracteriza-se por uma estrutura PaaS que permite descarregamento de tarefas computacionais intensas via um protocolo de comunicação baseado em *websockets*, proporcionando a comunicação dos *CellBots* com seus clones na nuvem.

A Figura 15 retrata uma visão simplificada do sistema apresentando como as camadas de comunicação estão organizadas, a disposição dos *CellBots* na rede local *ad hoc* e seus respectivos clones na nuvem robótica. Cada *CellBot* é representado de forma individual por retângulo. O retângulo maior representa a infraestrutura de nuvem com a plataforma *Rapyuta* com três clones em execução. Também é possível identificar que a linha tracejada representa a comunicação de cada *CellBot* com a nuvem (R2C) e a linha contínua a comunicão entre os robôs do MRS (R2R).

De forma geral, cada *CellBot*, no ambiente local, tem seu clone correspondente na nuvem. O clone é um conjunto de processos que funcionam em um ambiente computacional seguro. Toda comunicação entre os robôs e a nuvem é baseada em *websockets full duplex*, permitindo que tanto o clone quanto o *CellBot* enviem e recebam dados a qualquer momento, desde que a conexão esteja estabelecida previamente.

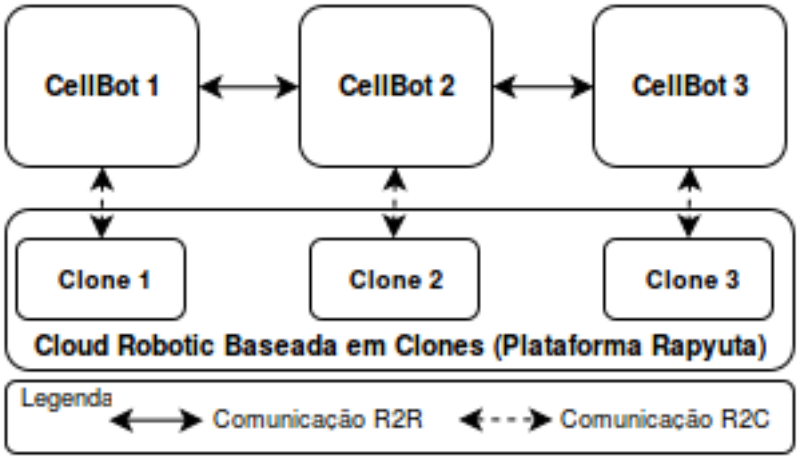

Figura 15 – Arquitetura Simplificada do Sistema.

Fonte: Próprio Autor.

4.1 CAMADA ROBOT-TO-ROBOT (R2R)

Na camada R2R, um grupo de robôs se comunicam através de conexões sem fio via tecnologia especificada pela IEEE 802.11(*Wi-Fi*), ideal para aplicações de curto alcance com demanda por largura de banda, formando uma malha de computação colaborativa e possibilitando a tomada de decisão por toda a equipe do MRS. Outra funcionalidade é permitir que robôs, que não estão em uma área de comunição com um ponto de acesso à nuvem, possam acessar informações armazenadas na infraestrutura de nuvem ou enviar solicitações de computação (entrega *multihop*). Para tal, é utilizado o protocolo AODV, ideal para redes altamente dinâmicas e de baixos recursos computacionais.

A Figura 16 retrata de forma detalhada as estruturas e a comunicação dentro dessa camada utilizando dois *CellBots*. Observa-se que cada *CellBot* é formado por três blocos principais, sendo dois deles diretamente relacionados aos recursos embarcados no *smartphone* adotado como unidade controladora: o sistema operacional, neste caso, o *Android*, e os próprios recursos de *hardware* do aparelho, com intuito de simplificar a ilustração só foram detalhadas as interfaces de comunicação, *bluetooth* e IEEE 802.11n. O terceiro bloco refere-se ao *Hardware Arduino*, sendo este responsável pelo sistema de locomoção do robô. Então, a comunicação entre esse sistema de locomoção e sua unidade controladora é feita através da tecnologia *bluetooth*, visto que não é necessário alta largura de banda e grande área de cobertura para tal função. Contrário a demanda de comunicação interna do *CellBot*, as necessidades da comunicação dentro da equipe de robôs é bem maior, pois exige uma área de cobertura abrangente, diante da capacidade de movimentação dos robôs e, ainda, alta largura de banda, diante da transmissão frequente de um volume considerável de dados, principalmente de imagens. Com isso, a interface adotada foi a 802.11n e para troca de mensagens o protocolo AODV.

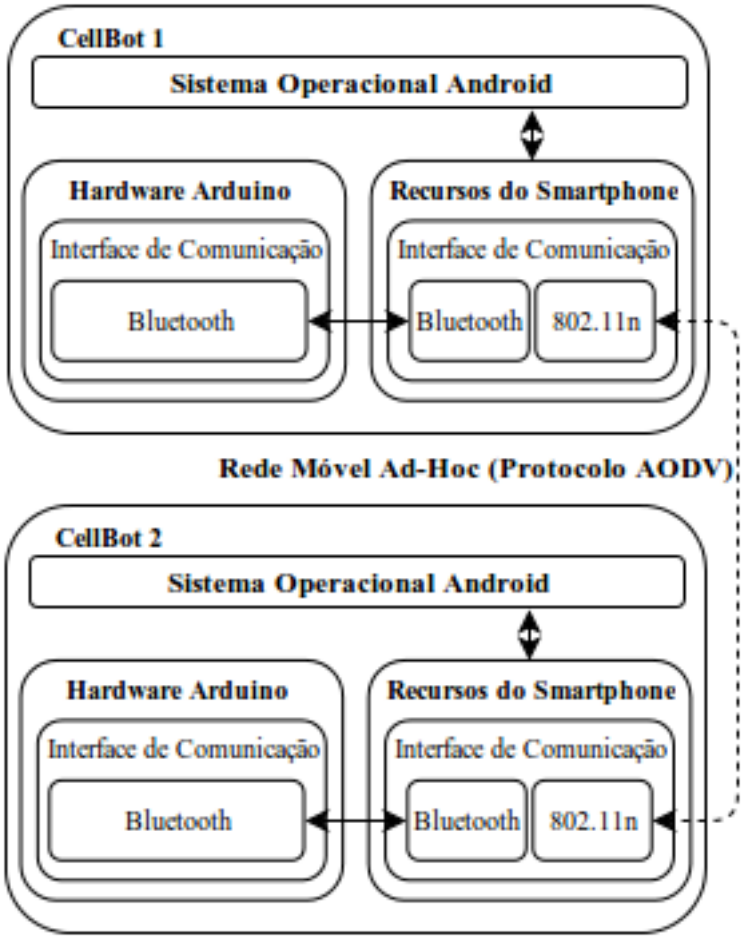

Figura 16 – Comunicação *Robot-to-Robot* via *interface* de rede IEEE 802.11n utilizando o protocolo de redes móveis *ad hoc* AODV.

Fonte: Próprio Autor.

4.1.1 Mensagens do Sistema Multirrobôs

Para manter a camada R2R funcionando de forma cooperativa, é necessários que os membros da equipe troquem mensagens para auxiliar na conclusão da missão. Para este trabalho, as mensagens utilizas pelos robôs são:

- *Robot Status* (RS): informa se o robô está livre ou realizando algum processamento de tarefa, além de garantir que o robô está *online* e incluso no MRS;
- *Task Status* (TS): traduz a situação da tarefa que o robô está executanto. Assume valores como: tarefa em execução, tarefa concluída e missão não concluída, tarefa concluída e missão concluída, e, por fim, tarefa abortada.

A mensagem *Robot Status* pode assumir os valores: (0), Robô Disponível: para indicar que o robô disponível para execução de uma tarefa; e (1), Robô Ocupado:

indicando a situação contrária, que o robô encontra-se executando uma tarefa naquele instante de tempo.

A mensagem *Task Status* pode assumir uma quantidade maior de valores tendo como referência a tarefa atribuída ao robô, sendo: (0), Tarefa em Execução: para indicar que, naquele instante de tempo, a tarefa atribuída ao robô está em execução e não foi concluída ainda; (1), Tarefa Concluída e Missão Não Concluída: significa que, apesar do robô concluir a tarefa atribuída a ele, a missão atribuída ao MRS não foi concluída até o instante de tempo que ele concluiu sua subtarefa. A partir desse momento o robô assume o estado Robô Disponível e pode vir a executar uma nova tarefa; (2), Tarefa Concluída e Missão Concluída: significa que o robô emissor dessa mensagem finalizou sua tarefa e, também, concluiu a missão. Ou seja, no caso de reconhecimento de objetos, ele localizou o objeto alvo; e, finalmente, (3), Tarefa Abortada: indica que por algum motivo o robô não concluiu sua tarefa. Um exemplo prático seria o fato de algum outro elemento da rede ter anunciado a mensagem Tarefa Concluída e Missão Concluída.

Essas informações adicionais são inseridas no cabeçalho do protocolo AODV através de um campo multidimensional e serializados em um *array* de *strings*. Como base para desenvolvimento da aplicação foram utilizados alguns trabalhos, como Jradi e Reedtz (2010) que desenvolveram uma aplicação que faz o papel de um protocolo de roteamento para redes *ad hoc* dentro do sistema operacional Android. E ainda, Júnior (2014) que desenvolveu um aplicativo para atribuição automática de endereços IP (*Internet Protocol*) em uma rede *ad hoc* e Machado e Melo (2014) que realizaram adapatações na métrica do AODV, considerando a carga da bateria e não a quantidade de saltos para gerar o custo das rotas dento da MANET.

4.1.2 Caso de Uso

Com objetivo de exemplificar a utilização das mensagens *Robot Status* (RS) e *Task Status* (TS) é apresentado um caso de uso a partir de um MRS formado por três *CellBots*. Cada *CellBot* é identificado por um número inteiro único dentro do MRS e será representado como um retângulo. Para simplificar a ilustração, em cada instante de tempo, o qual vai variar entre 0, valor inicial, e 3, valor final, é apenas exibido os valores das mensagens RS e TS no formato [RS][TS]. Logo, para cada instante de tempo é apresentado os valores que cada *CellBot* assumi até a missão atribuída ao MRS ser concluída.

A Figura 17 representa o estado inicial do MRS, portanto, na grandeza tempo trata-se como t_0. Neste momento, tem-se todos os *CellBots* com os mesmo valores, sendo RS=0 e TS=0. Significa que os robôs do MRS estão operacionais e disponíveis para

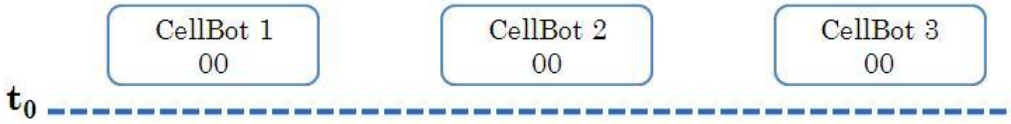

Figura 17 – Mensagens RS e TS no instante de tempo t_0 para cada *CellBot* do MRS.

Fonte: Próprio Autor.

execução de uma nova tarefa.

A partir do instante de tempo t_1, apresentado na Figura 18, os *CellBots* mudam seu estado de disponível para ocupado, ou seja, RS=1. Isso significa que estão executando uma determinada missão atribuída a eles. Acrecenta-se que TS=0 indica que nesse instante de tempo a tarefa atribuída ao robô encontra-se em execução.

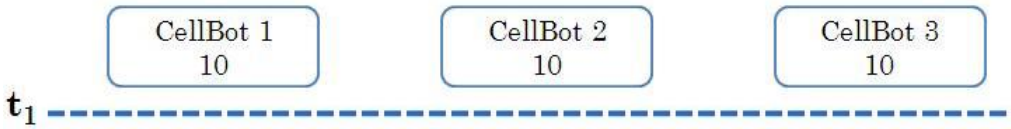

Figura 18 – Mensagens RS e TS no instante de tempo t_1 para cada *CellBot* do MRS.

Fonte: Próprio Autor.

Já no instante de tempo t_2, cada *CellBot* apresenta uma saída diferente. Onde o *CellBot* 1 tem RS=1 e TS=1, significa que ele completou sua tarefa, no entanto, a missão global atribuída ao MRS não foi concluída. Por outro lado, o *CellBot* 2 tem RS=1 e TS=2, ou seja, o robõ foi capaz de finalizar a tarefa atribuída a ele e, ainda, conseguiu concluir a missão atribuída ao MRS. Considerando um exemplo de busca por um determinado objeto, o *CellBot* 2 teria feito a localização do mesmo. Porém, o *CellBot* 3, ao receber a mensagem publicada pelo *CellBot* 2, imediatamente aborta sua missão, TS=3, visto que a missão atribuída ao MRS foi concluída, não sendo mais necessário gastar recursos com uma missão concluída.

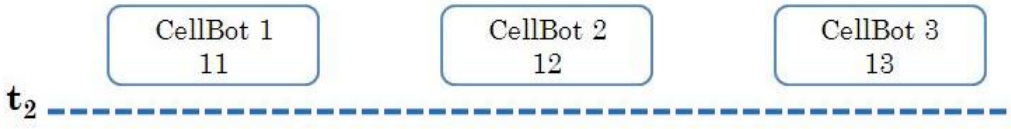

Figura 19 – Mensagens RS e TS no instante de tempo t_2 para cada *CellBot* do MRS.

Fonte: Próprio Autor.

Por fim, no instante de tempo t_3 (Figura 20), todos os *CellBots* do MRS retornam ao estado inicial com valores RS=0 e TS=0, para aguardar a atribuição de uma nova missão.

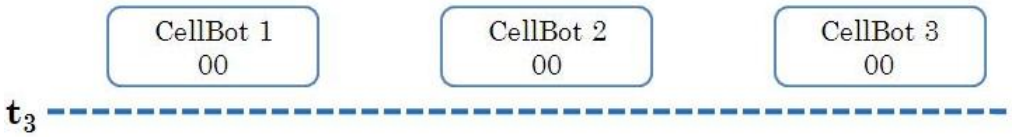

Figura 20 – Mensagens RS e TS no instante de tempo t_3 para cada *CellBot* do MRS.

Fonte: Próprio Autor.

4.2 CAMADA ROBOTIC-TO-CLOUD (R2C)

Na camada R2C, a infraestrutura de nuvem física fornece um conjunto de recursos computacionais e de armazenamento que podem ser alocados elasticamente para a demanda em tempo real da equipe de *CellBots*, retirando a necessidade dos robôs de armazenamento ou processamento de uma grande quantidade de dados. Um dos benefícios proporcionado por este armazenamento é permitir a condensação de um grande volume de dados, que, quando processados, resultam em informações importantes para uso de todo o MRS. Um exemplo dessa aplicação é a localização de obstáculos dentro do mapa. Outros benefícios já foram citados, como a redução do consumo de energia e a utilização de robôs de baixo custo, visto que os recursos estão disponíveis na nuvem e não mais alocados a bordo do robô.

A utilização da plataforma *Rapyuta* permite que cada *CellBot*, após autenticado, crie um clone que será responsável por executar as tarefas solicitadas, bem como devolver o resultado delas para robô dentro do MRS. Para acessar a plataforma, o *CellBot* faz uso da tecnologia *Websocket* implementado via ferramentas do projeto *Autobahn*[1] que é executado sobre o *framework Twisted*[2]. Isto permite uma única conexão bidirecional (*full duplex*) com o robô, logo, a *Rapyuta* pode enviar dados para o cliente a qualquer momento, desde que a conexão esteja previamente estabelecida. A Figura 21 esquematiza essa comunicação.

4.3 SERVIÇOS DE PLANEJAMENTO DE CAMINHO E RECONHECIMENTO DE OBJETOS

Cada *CellBot*, devidamente autenticado na plataforma, conforme descrito anteriormente, tem um clone na nuvem. É necessário ficar claro que o termo clone faz referência a um conjunto de processos utilizados para fornecer os serviços ao robô no seu ambiente local. Com isso, o processo responsável em fazer intermédio entre o robô real e a nuvem é o *Robot EndPoint* (*Robot* EP). Ele é equipado com interfaces e portas que

[1] http://autobahn.ws/android/
[2] https://twistedmatrix.com/trac/

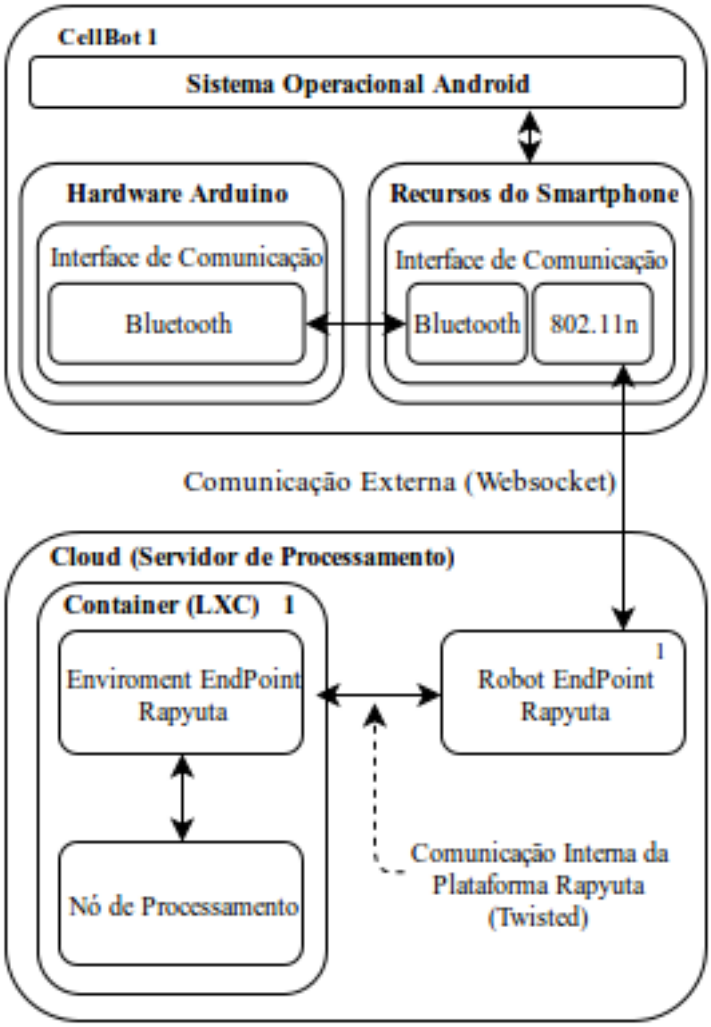

Figura 21 – Comunicação *Robot-to-Cloud* via *interface* de rede IEEE 802.11n utilizando *websocket* para acesso à plataforma *Rapyuta*.

Fonte: Próprio Autor.

conectam o *CellBot* ao *Environment EndPoint* (*Envinronment* EP). Os *Environments EP*, por sua vez, são ambientes de computação da *Rapyuta* implementados com a tecnologia *Linux Containers* (LXC) compatíveis com o ROS. É configurado para executar qualquer nó de processamento ROS, além de viabilizar a comunicação entre esses nós e, caso necessário, de cada nó com o *Robot EndPoint*. Para saber mais sobre a plataforma *Rapyuta* e detalhes de sua comunicação interna, recomenda-se fortemente a leitura de Hunziker et al. (2013) e Mohanarajah et al. (2015a).

Logo, para disponibilizar os serviços de planejamento de caminho e reconhecimento de objetos para os *CellBots*, é necessário criar pacotes ROS. Para tal, todo código fonte para as serviços disponíveis na nuvem foi feito na linguagem de programação C++, utilizando a biblioteca *OpenCV* 2. Com isso, qualquer robô pode usufruir desta arquitetura, visto que não há restrição de *hardware*, sendo apenas necessário *interface* de comunicação IEEE 802.11n e uma câmera para capturar imagens para lançar à nuvem.

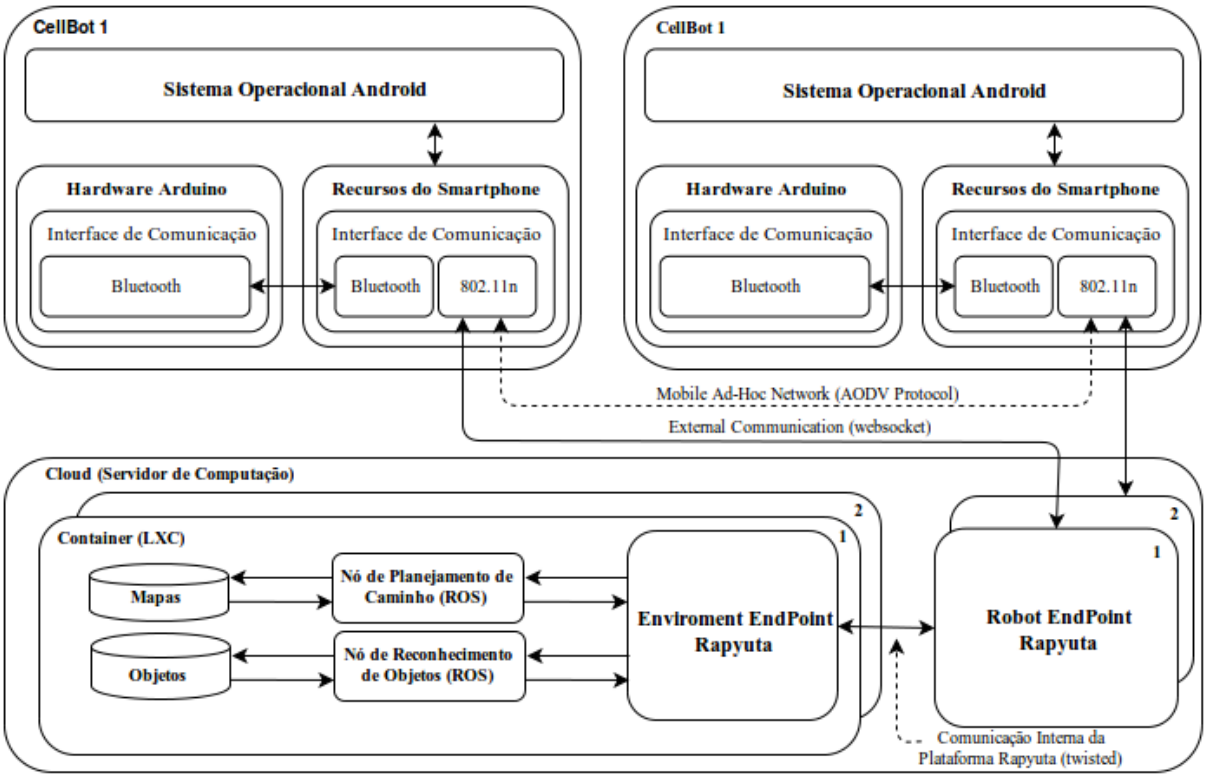

Figura 22 – Comunicação *Robot-to-Cloud* via *interface* de rede IEEE 802.11n utilizando *websocket* para acesso a plataforma *Rapyuta*.

Fonte: Próprio Autor.

De forma geral o funcionamento do sistema, esquematizado na Figura 22, pode ser descrito nas seguintes etapas:

1. Formação da rede móvel *ad hoc* pelos *CellBots* com a condição de que, no mínimo, um nó tenha conectividade com a plataforma *Rapyuta*;

2. Cada robô da MANET cria seu representante na nuvem para poder ter acesso aos serviços ofertados, como planejamento de caminho e reconhecimento de objetos;

3. O robô clone envia para o *CellBot* as informações sobre o trajeto que ele deve seguir a partir do mapa disponibilizado no banco de dados e a posição inicial do robô real, que é enviada no momento de conexão com o clone;

4. O *CellBot*, ao chegar ao ponto marcado para busca, utiliza sua câmera para buscar a imagem previamente cadastrada no banco de imagens, afinal o objetivo da missão é localizar um objeto dentro do mapa;

5. Após buscar no local demarcado, o robô retorna ao ponto inicial, independente de ter localizado ou não o objeto. Porém, aquele robô que teve êxito na busca comunica aos demais membros da equipe o ocorrido, a posição na qual o objeto se encontra, por fim, que a tarefa foi concluída.

5 RESULTADOS

Com intuito de realizar verificações sobre o funcionamento da plataforma e seu comportamento na infraestrutura do LAR-UERN, foi montada uma topologia semelhante à presente em Mohanarajah et al. (2015a) para verificar a viabilidade de empregá-la como *Cloud Robotics* para os *CellBots*. Nesse sentido, foram feitas medições de RTT (*Round Trip Times*) (KARN; PARTRIDGE, 1987) com mensagens de diferentes tamanhos entre processos com variação em sua localização dentro da arquitetura *Rapyuta*. Essa metodologia de testes foi adotada por Hunziker et al. (2013) e Mohanarajah et al. (2015a), e adaptada em DePalma e Breazeal (2016) e Salmerón-García et al. (2016), com objetivo de retratar o comportamento da plataforma nos mais variados cenários, sempre dando destaque aos componentes envolvidos no processo de comunicação e os elementos que estão relacionados ao seu desempenho.

Para cada configuração foram feitas 35 amostras de 25 tamanhos de mensagens variando de 10 Bytes a 10 MB. A infraestrutura de nuvem foi formada por três máquinas de mesma capacidade, sendo CPU AMD Athlon II, memória principal 3,5 Gb e sistema operacional Ubuntu 12.04 Precise. A rede local entre essas máquinas é formado por cabos categoria 5e, taxa de 100 Mbps. Já o robô utilizado conectou-se via IEEE 802.11n.

5.1 CENÁRIOS DE TESTE

O cenário criado para tais testes foi formado por um servidor responsável em receber e despachar as requisições (Servidor de Computação 1), e dois outros servidores (Servidores de Computação 2 e 3), responsáveis por hospedar os ambientes de computação (*containers*). Cada Servidor de Computação continha dois *containers*, os quais, por sua vez, tinham em execução dois nós ROS, *TestRunner* e *StringEcho*. O nó *TestRunner* era encarregado de criar e enviar as mensagens de variados tamanhos, bem como, a captura do tempo de viajem das mesmas (RTTs). Já o nó *StringEcho*, por outro lado, foi incubido de receber as mensagens enviadas pelo *TestRunner* e devolvê-las. A partir disso, foram realizados testes em quatro cenários distintos, variando os componentes envolvidos e sua localização dentro da infraestrutura de nuvem. Os cenários utilizados são apresentados a seguir.

5.1.1 Node-to-Node (N2N)

Neste primeiro cenário, trabalhou-se com a comunicação entre os nós ROS em execução no mesmo ambiente de computação (*container*) e, consequentemente, no mesmo servidor de computação, logo existe uma quantidade mínima de elementos intermediários no processo de envio e recebimento das mensagens, afinal o processamento ocorre dentro da mesma máquina virtual. A Figura 23 retrata com detalhes o cenário N2N.

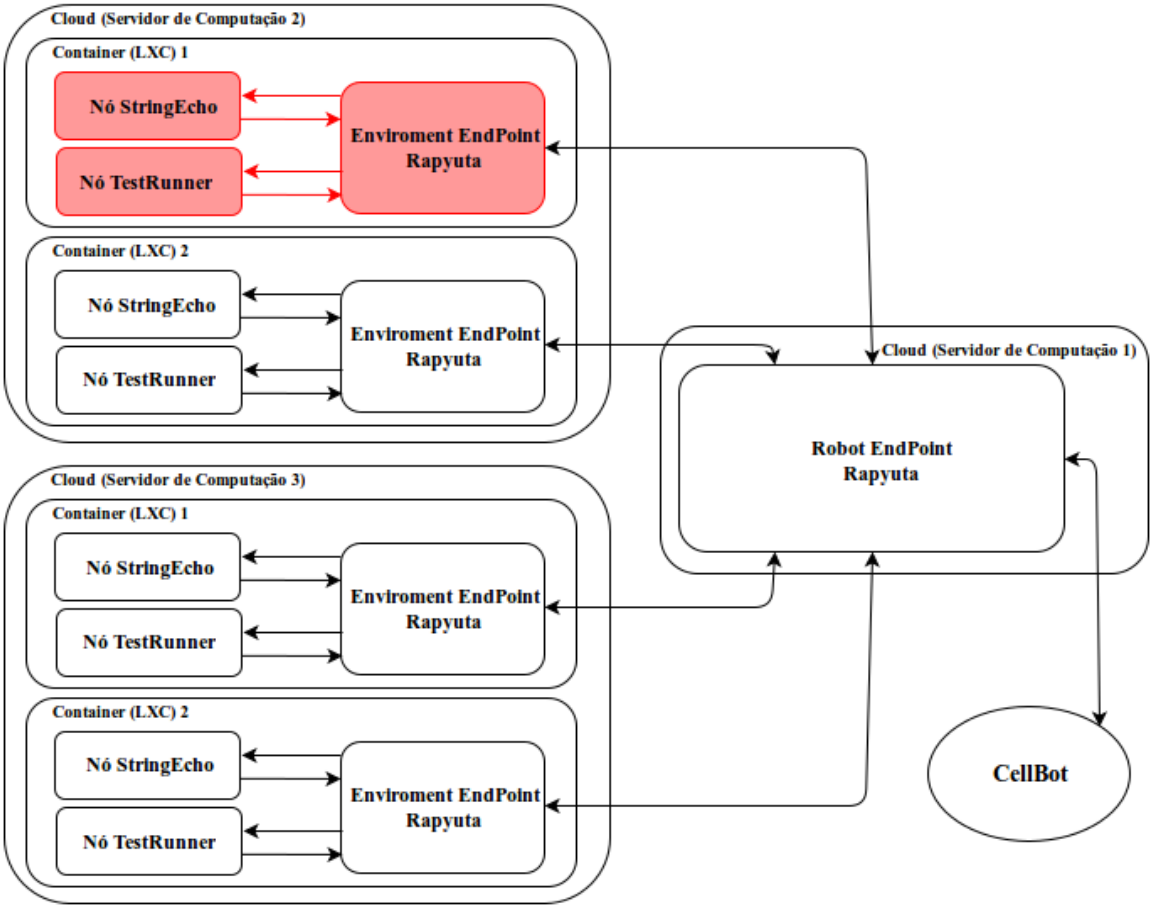

Figura 23 – Configuração da plataforma *Rapyuta* para comunicação *Node-to-Node* (N2N).

Fonte: Próprio Autor.

5.1.2 Container-to-Container 1 (C2C-1)

O formato do segundo cenário, apresentado na Figura 24, diferencia do cenário anterior na localização dos nós, visto que o nó *TestRunner* é executado no *container* 1 e o nó *StringEcho* no *container* 2, porém no mesmo servidor de computação. Com isso, é necessário que novos elementos intervenham no processo de comunicação, como, por exemplo, o *Robot EndPoint*, afinal todos os ambientes de computação criados são associados a um robô virtual instanciado na nuvem.

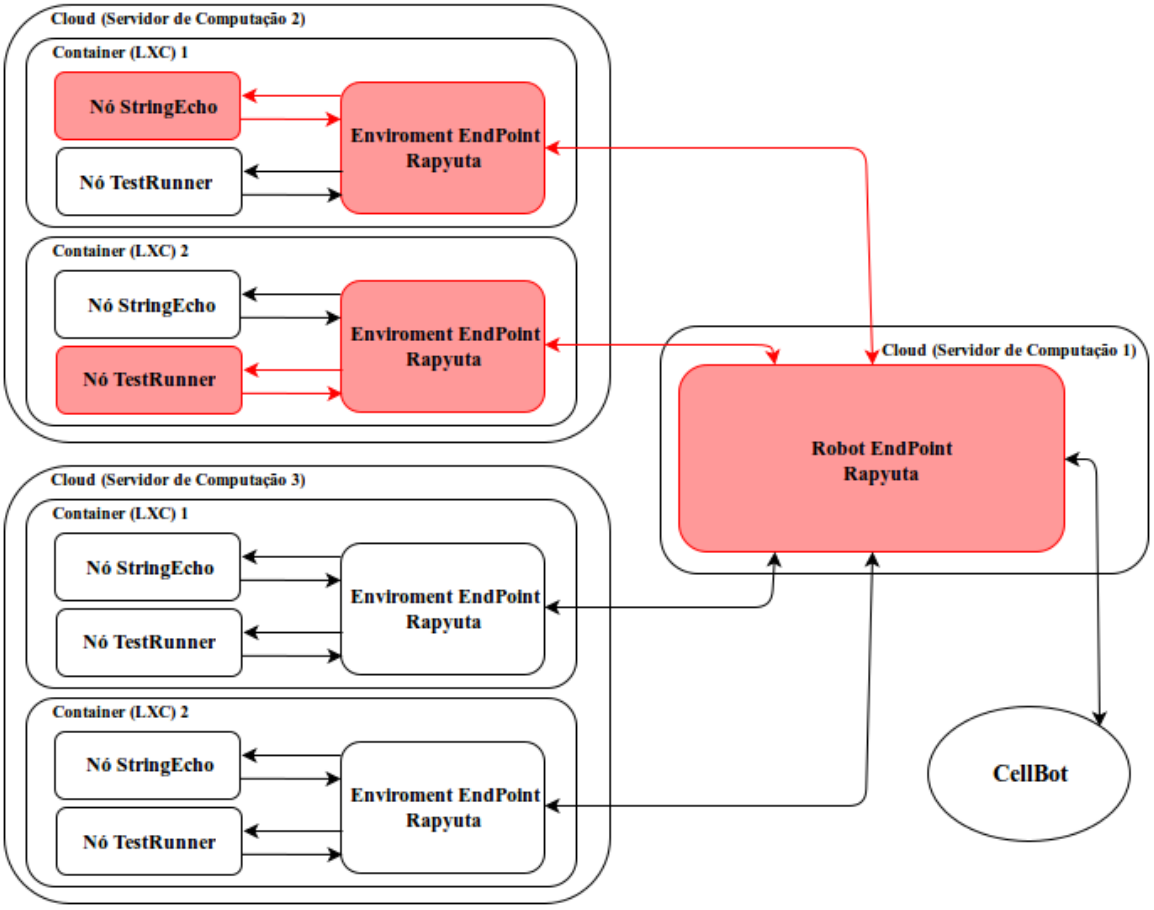

Figura 24 – Configuração da plataforma *Rapyuta* para comunicação *Container-to-Container 1* (C2C-1).

Fonte: Próprio Autor.

5.1.3 Container-to-Container 2 (C2C-2)

A Figura 25 retrata o terceiro cenário, o qual diferencia-se do anterior devido a utilização dos dois servidores de computação, servidores 2 e 3, os quais hospedaram os nós *StringEcho* e *TestRunner*, respectivamente. Esse formato é considerado o mais complexo, visto que cada servidor de computação utilizado instancia *containers* para hospedar os nós ROS. Com isso, é esperado um tempo maior para troca de dados, diante da existência de uma quantidade maior de elementos intermediários.

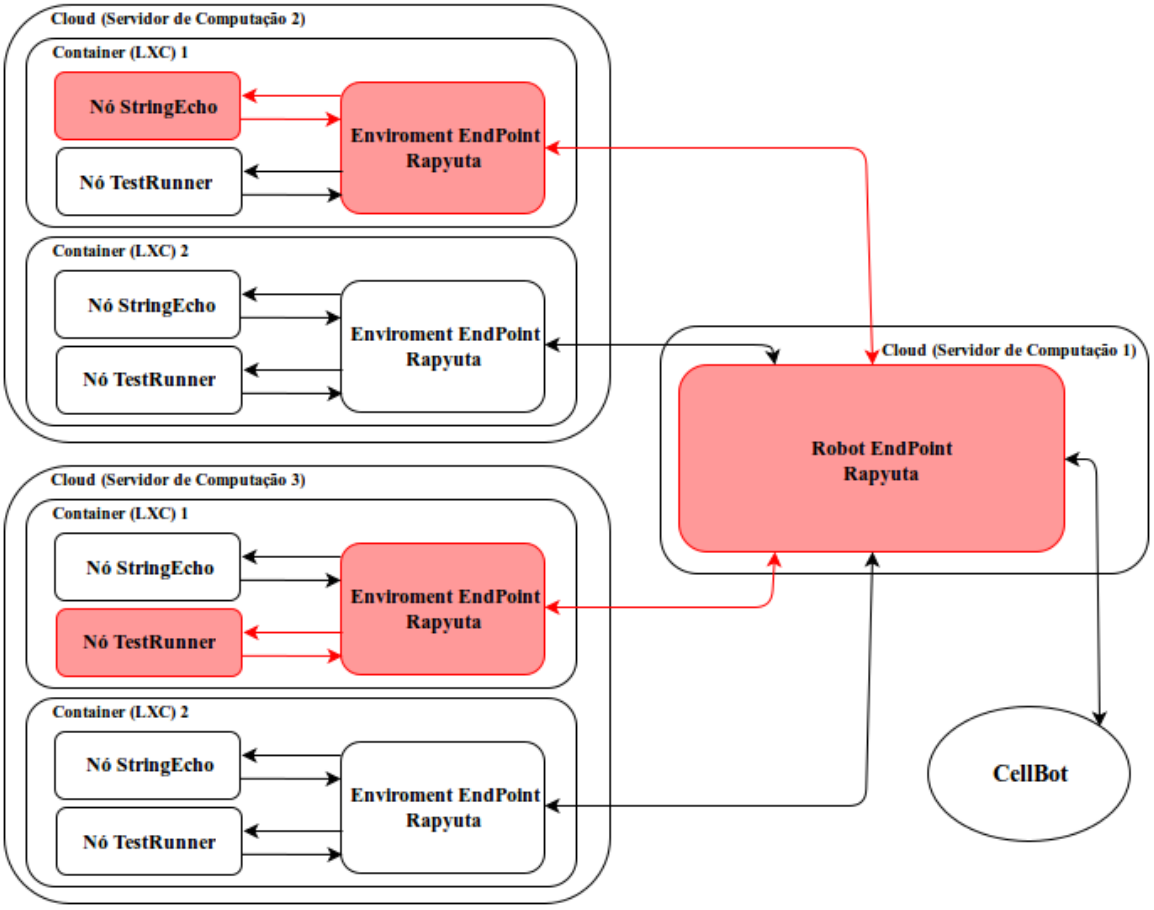

Figura 25 – Configuração da plataforma *Rapyuta* para comunicação *Container-to-Container 2* (C2C-2).

Fonte: Próprio Autor.

5.1.4 Robot-to-Cloud (R2C)

Por fim, o quarto cenário ilustra a comunicação entre um robô (*CellBots*) e a nuvem, onde o serviço utilizado pelo robô está hospedado em um servidor de computação, o qual executa um nó ROS *StringEcho* em um ambiente de computação (*container*). É esperado um valor maior para RTT, pois a comunicação entre o *CellBot* e a plataforma de *Cloud Robotics* é feita via interface de comunicação IEEE 802.11n, ou seja, sem fio, enquanto os demais testes só utilizaram como enlace de comunicação cabos metálicos cat5e, o que melhora o desempenho de qualquer processo de comunicação.

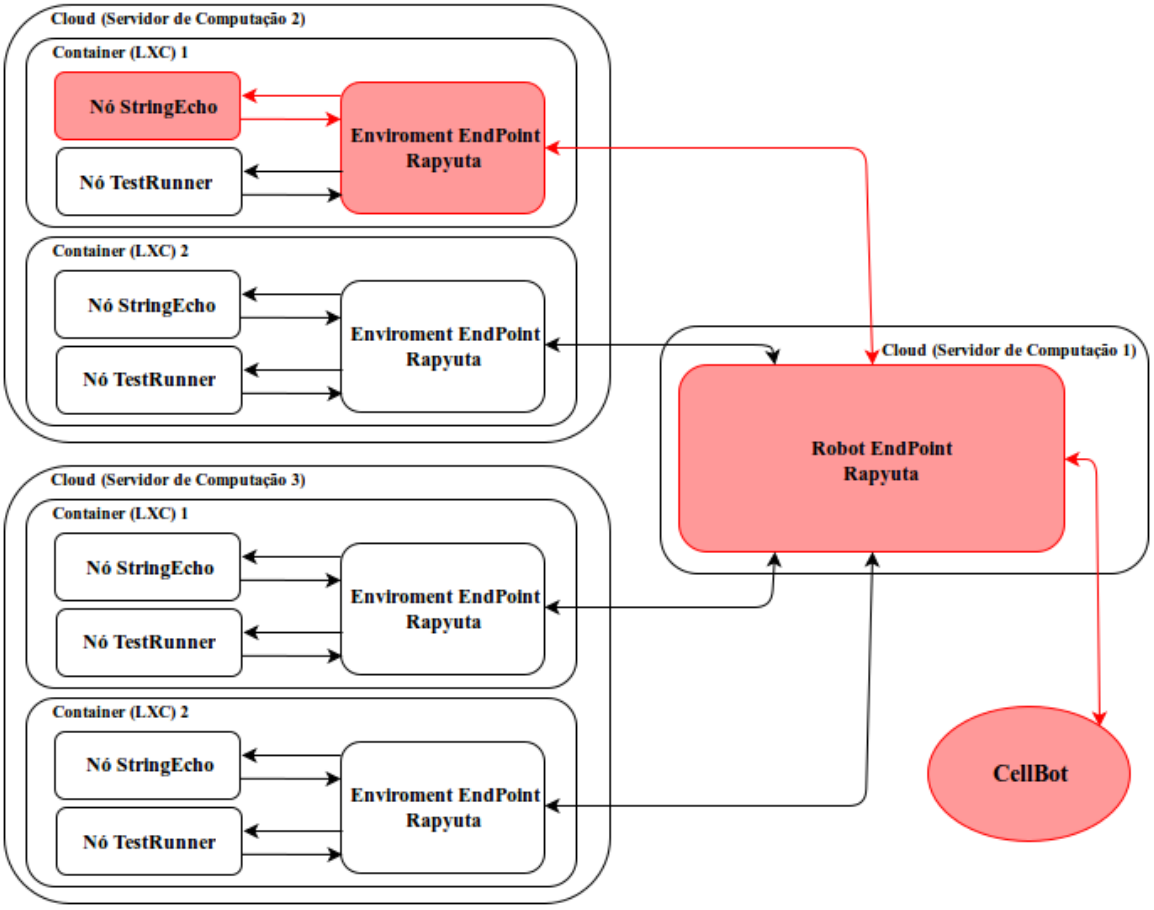

Figura 26 – Configuração da plataforma *Rapyuta* para comunicação *Robot-to-Cloud* (R2C).

Fonte: Próprio Autor.

5.2 RESULTADOS

A Figura 27 apresenta os resultados obtidos com os testes e evidencia algumas características da arquitetura. Para mensagens de até 10 KB ocorre pouca variação em todos os formatos, porém na comunicação *Robot-to-Cloud* (R2C) fica claro não ser interessante trabalhar com mensagens tão grandes, visto que o tempo de RTT é, relativamente, alto, quando comparado com os demais formatos. Isso pode comprometer o funcionamento do MRS, caso o tempo de resposta oriunda da nuvem de computação tenha influência direta no resultado da missão.

Além disso, vale ressaltar a diferença na comunicação entre *containers*, sendo C2C-1, *containers* instanciados na mesma máquina, e C2C-2, em máquinas diferentes. Fica claro o atraso ocasionado com o encaminhamento de pacotes entre as redes virtuais e real. Por fim, o melhor desempenho foi apresentado pela comunicação *Node-to-Node* (N2N), evidenciando que, para conseguir extrair o melhor desempenho da plataforma, é interessante manter os nós ROS executando no mesmo ambiente de computação (*container*).

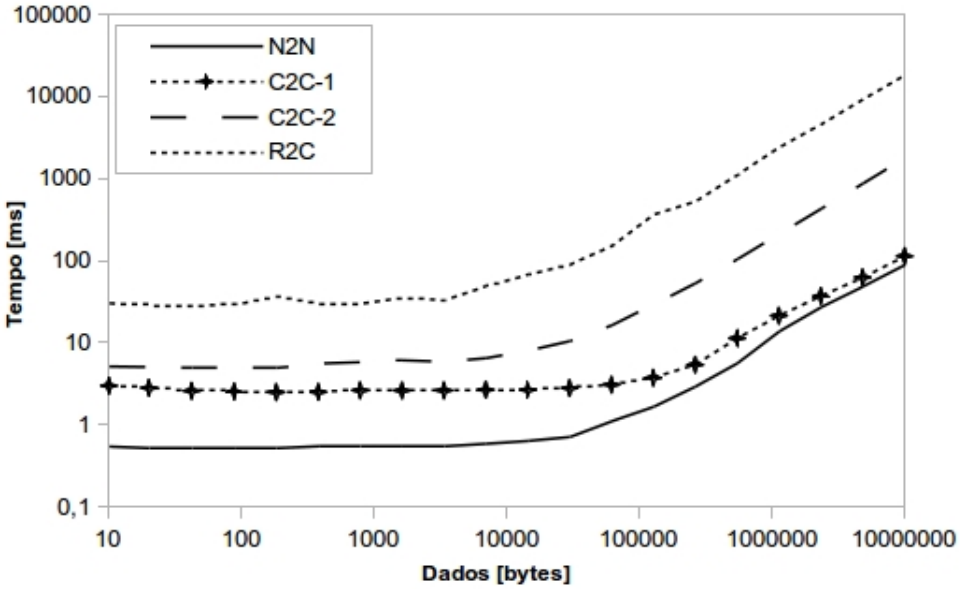

Figura 27 – RTTs para diferentes configurações da plataforma Rapyuta variando conforme os componentes envolvidos na comunicação.

Fonte: Próprio Autor.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Foi apresentada uma arquitetura de comunicação para prover robótica na nuvem para uma equipe de *CellBots*. Foi feito um detalhamento dos componentes envolvidos e como estão organizados para que isso seja possível. Também foi realizado testes de medição de RTT para comprovar a viabilidade na transmissão de grandes mensagens de dados. Os resultados obtidos são satisfatórios para atender as primeiras demandas que são envios de imagens e posição do robô dentro do mapa. As principais vantagens de trabalhar com a plataforma *Rapyuta* foi ter ambientes de computação alocados dinamicamente pelos próprios robôs do MRS e, ainda, trabalhar com o ROS, permitindo que diferentes tipos de robôs possam fazer uso dos serviços desenvolvidos.

Para o futuro pretende-se realizar um aprofundamento na área de *Cloud Robotics* e na plataforma *Rapyuta* para desenvolver uma infraestrutura de nuvem própria, que também seja capaz ofertar os benefícios de *Cloud Computing* para robôs. Além disso, também é de grande interesse seguir nas pesquisas com *CellBots* tendo em vista a grande variedade de aplicações que podem ser desenvolvidas para esses dispositivos, principalmente por trabalhar com a platafor Android.

Uma série de itens podem ser explorador com maior ênfase em trabalhos futuros, dentre eles pode-se destacar realizações de simulações no *Network Simulator*-3 com objetivo de verificar a escalabilidade do sistema utilizando métricas distintas para o protocolo AODV. Também pode-se realizar comparações com processamento de aplicações a bordo do robô e na nuvem, para verificar em números o real benefício de prover *Cloud Robotics* para esses dispostivos.

Cloud Robotics permite que robôs possam compartilhar recursos computacionais, informações e dados entre si, e, ainda, acessar novos conhecimentos e habilidades as quais ainda não aprendeu. Isso abre um novo paradigma na área de robótica que acredita-se que conduz a desenvolvimentos futuros interessantes. Permite o envio de robôs mais baratos e com menor consumo de energia, de computação e de memória, aproveitando as redes de comunicações e os recursos computacionais elásticos oferecidos pela infraestrutura de nuvem. Diversas aplicações já estão sendo desenvolvidas com esse modelo e apresentam resultados significativos, como por exemplo o SLAM, e muitos outros que não foram discutidas, como monitoramento meteorológico, detecção de intrusão, vigilância e controle de formação. Sabe-se que pesquisas nessa área são promissoras e, além disso, podem facilmente enquadrar-se no dia a dia das cidades, permitindo à academia promover qualidade de vida.

REFERÊNCIAS

ABOLHASAN, M.; WYSOCKI, T.; DUTKIEWICZ, E. A review of routing protocols for mobile ad hoc networks. *Ad hoc networks*, Elsevier, v. 2, n. 1, p. 1–22, 2004. 28 Citado na página 28.

ALLIANCE, Z. et al. *Zigbee specification*. 2006. 31 Citado na página 31.

AMBROSIA, V. G. et al. Demonstrating uav-acquired real-time thermal data over fires. *Photogrammetric engineering & remote sensing*, American Society for Photogrammetry and Remote Sensing, v. 69, n. 4, p. 391–402, 2003. 14 Citado na página 14.

AROCA, R. V. et al. Increasing students' interest with low-cost cellbots. *Education, IEEE Transactions on*, IEEE, v. 56, n. 1, p. 3–8, 2013. 20 Citado na página 20.

AROCA, R. V. et al. Towards smarter robots with smartphones. In: *5th Workshop in Applied Robotics and Automation, Robocontrol*. [S.l.: s.n.], 2012. 17, 21, 22 Citado 3 vezes nas páginas 17, 21, and 22.

ARTHUR, M. B. J. *Automatic control system for vehicles*. [S.l.]: Google Patents, 1966. US Patent 3,245,493. 13 Citado na página 13.

ARUMUGAM, R. et al. Davinci: A cloud computing framework for service robots. In: IEEE. *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. [S.l.], 2010. p. 3084–3089. 43 Citado na página 43.

BELLINGHAM, J. G.; RAJAN, K. Robotics in remote and hostile environments. *Science*, American Association for the Advancement of Science, v. 318, n. 5853, p. 1098–1102, 2007. 13 Citado na página 13.

CAO, Y. U.; FUKUNAGA, A. S.; KAHNG, A. Cooperative mobile robotics: Antecedents and directions. *Autonomous robots*, Kluwer Academic Publishers, v. 4, n. 1, p. 7–27, 1997. 20 Citado na página 20.

CLARK, A.; GRUSH, R. Towards a cognitive robotics. *Adaptive Behavior*, Sage Publications, v. 7, n. 1, p. 5–16, 1999. 16 Citado na página 16.

CLAUSEN, T.; JACQUET, P. *Optimized link state routing protocol (OLSR)*. [S.l.], 2003. 29 Citado na página 29.

CORSON, S.; MACKER, J. *Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations*. [S.l.], 1998. 27 Citado na página 27.

CROW, B. P. et al. Ieee 802.11 wireless local area networks. *Communications Magazine, IEEE*, IEEE, v. 35, n. 9, p. 116–126, 1997. 23 Citado na página 23.

DEPALMA, N.; BREAZEAL, C. Nimbus: A hybrid cloud-crowd realtime architecture for visual learning in interactive domains. *arXiv preprint arXiv:1602.07641*, 2016. 41, 53 Citado 2 vezes nas páginas 41 and 53.

FENWICK, J. W.; NEWMAN, P. M.; LEONARD, J. J. Cooperative concurrent mapping and localization. In: IEEE. *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. [S.l.], 2002. v. 2, p. 1810–1817. 15 Citado na página 15.

FERNANDES, A. et al. Wirelessteams: Comparação de tecnologias sem fios em equipas de robôs móveis. *Relatório técnico, Coimbra*, 2012. 23 Citado na página 23.

FERNANDO, N.; LOKE, S. W.; RAHAYU, W. Mobile cloud computing: A survey. *Future Generation Computer Systems*, Elsevier, v. 29, n. 1, p. 84–106, 2013. 22 Citado na página 22.

FOX, D. et al. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, IEEE, v. 94, n. 7, p. 1325–1339, 2006. 15 Citado na página 15.

GOYAL, P.; PARMAR, V.; RISHI, R. Manet: vulnerabilities, challenges, attacks, application. *IJCEM International Journal of Computational Engineering & Management*, v. 11, n. 2011, p. 32–37, 2011. 28 Citado na página 28.

GUIZZO, E.; DEYLE, T. Robotics trends for 2012. *IEEE Robotics & Automation Magazine*, v. 19, n. 1, p. 119–123, 2012. 17 Citado na página 17.

HAAS, Z. J. et al. Wireless ad hoc networks. *Encyclopedia of Telecommunications*, Wiley Online Library, 2002. 25 Citado na página 25.

HAAS, Z. J.; PEARLMAN, M. R.; SAMAR, P. The zone routing protocol (zrp) for ad hoc networks. 2002. 29 Citado na página 29.

HE, G. Destination-sequenced distance vector (dsdv) protocol. *Networking Laboratory, Helsinki University of Technology*, p. 1–9, 2002. 29 Citado na página 29.

HEIDEMANN, J. et al. Effects of detail in wireless network simulation. In: *Proceedings of the SCS multiconference on distributed simulation*. [S.l.: s.n.], 2001. p. 3–11. 28 Citado na página 28.

HU, G.; TAY, W. P.; WEN, Y. Cloud robotics: architecture, challenges and applications. *Network, IEEE*, IEEE, v. 26, n. 3, p. 21–28, 2012. 15, 17, 32, 35, 42, 44, 45 Citado 7 vezes nas páginas 15, 17, 32, 35, 42, 44, and 45.

HUNZIKER, D. et al. Rapyuta: The roboearth cloud engine. In: IEEE. *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. [S.l.], 2013. p. 438–444. 39, 43, 45, 51, 53 Citado 5 vezes nas páginas 39, 43, 45, 51, and 53.

JOHNSON, D. B. et al. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad hoc networking*, v. 5, p. 139–172, 2001. 29 Citado na página 29.

JORDAN, S. et al. The rising prospects of cloud robotic applications. In: IEEE. *Computational Cybernetics (ICCC), 2013 IEEE 9th International Conference on*. [S.l.], 2013. p. 327–332. 16, 22, 35, 37, 38 Citado 5 vezes nas páginas 16, 22, 35, 37, and 38.

JRADI, R. K.; REEDTZ, L. S. *Ad-hoc network on Android*. Tese (Doutorado) — Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2010. 48 Citado na página 48.

JÚNIOR, E. d. C. O. Atribuição dinâmica de endereços em redes ad hoc sem fio em plataforma android. 2014. 31, 48 Citado 2 vezes nas páginas 31 and 48.

KARN, P.; PARTRIDGE, C. Improving round-trip time estimates in reliable transport protocols. *ACM SIGCOMM Computer Communication Review*, ACM, v. 17, n. 5, p. 2–7, 1987. 53 Citado na página 53.

KHALIL, W.; DOMBRE, E. *Modeling, identification and control of robots*. [S.l.]: Butterworth-Heinemann, 2004. 13 Citado na página 13.

KOH, L.; WICH, S. Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation. Mongabay. com, 2012. 14 Citado na página 14.

KOKEN, B. Cloud robotics platforms. *Interdisciplinary Description of Complex Systems*, Hrvatsko interdisciplinarno društvo, v. 13, n. 1, p. 26–33, 2015. 36, 37 Citado 2 vezes nas páginas 36 and 37.

KOUVATSOS, D. D.; MISKEEN, G. M. Performance related security modelling and evaluation of ranets. *Wireless Personal Communications*, Springer, v. 64, n. 3, p. 523–546, 2012. 32 Citado na página 32.

LEE, A.; RA, I.; KIM, H. Performance study of ad hoc routing protocols with gossip-based approach. In: SOCIETY FOR COMPUTER SIMULATION INTERNATIONAL. *Proceedings of the 2009 Spring Simulation Multiconference*. [S.l.], 2009. p. 89. 29 Citado na página 29.

LEE, J.-S.; SU, Y.-W.; SHEN, C.-C. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. In: IEEE. *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*. [S.l.], 2007. p. 46–51. 25 Citado na página 25.

LIU, J. N.; WANG, M.; FENG, B. ibotguard: an internet-based intelligent robot security system using invariant face recognition against intruder. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, v. 35, n. 1, p. 97–105, 2005. 13 Citado na página 13.

MACHADO, H. P.; MELO, R. A. S. de. Protocolo aodv com eficiência energética para plataforma android. 2014. 48 Citado na página 48.

MANYIKA, J. et al. Big data: The next frontier for innovation, competition, and productivity. 2011. 16 Citado na página 16.

MARCHANT, G. E. et al. International governance of autonomous military robots. *Colum. Sci. & Tech. L. Rev.*, v. 12, p. 272–276, 2011. 13 Citado na página 13.

MELL, P.; GRANCE, T. The nist definition of cloud computing: Recommendations of the national institute of standards and technology (2011). *URL http://csrc. nist. gov/publications/nistpubs/800-145/SP800-145. pdf*, 2012. 16 Citado na página 16.

MESTER, G. Cloud robotics model. *Interdisciplinary Description of Complex Systems*, Hrvatsko interdisciplinarno društvo, v. 13, n. 1, p. 1–8, 2015. 38 Citado na página 38.

MICHAEL, N. et al. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, Wiley Online Library, v. 29, n. 5, p. 832–841, 2012. 15 Citado na página 15.

MOHANARAJAH, G. et al. Rapyuta: A cloud robotics platform. *Automation Science and Engineering, IEEE Transactions on*, IEEE, v. 12, n. 2, p. 481–493, 2015. 19, 40, 42, 45, 51, 53 Citado 6 vezes nas páginas 19, 40, 42, 45, 51, and 53.

MOHANARAJAH, G. et al. Cloud-based collaborative 3d mapping in real-time with low-cost robots. *IEEE Transactions on Automation Science and Engineering*, IEEE, v. 12, n. 2, p. 423–431, 2015. 43 Citado na página 43.

MONTEMERLO, M. et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In: *Aaai/iaai*. [S.l.: s.n.], 2002. p. 593–598. 43 Citado na página 43.

MURPHY, R. R. Rescue robotics for homeland security. *Communications of the ACM*, ACM, v. 47, n. 3, p. 66–68, 2004. 13 Citado na página 13.

OSUNMAKINDE, I.; RAMHARUK, V. Development of a survivable cloud multi-robot framework for heterogeneous environments. *International Journal Advanced Robotic Systems*, v. 11, p. 164, 2014. 17, 27, 33, 34, 38, 42, 44, 45 Citado 8 vezes nas páginas 17, 27, 33, 34, 38, 42, 44, and 45.

PANAIT, L.; LUKE, S. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, Kluwer Academic Publishers, v. 11, n. 3, p. 387–434, 2005. 16 Citado na página 16.

PAREDIS, C. et al. Millibots: The development of a framework and algorithms for a distributed heterogeneous robot team. Georgia Institute of Technology, 2002. 20 Citado na página 20.

PERKINS, C.; BELDING-ROYER, E.; DAS, S. *Ad hoc on-demand distance vector (AODV) routing*. [S.l.], 2003. 29, 30, 44 Citado 3 vezes nas páginas 29, 30, and 44.

ROCHA, R. P. P. d. Building volumetric maps with cooperative mobile robots and useful information sharing: a distributed control approach based on enthropy. 2006. 13, 14, 15 Citado 3 vezes nas páginas 13, 14, and 15.

SALMERÓN-GARCÍA, J. et al. Study of communication issues in dynamically scalable cloud-based vision systems for mobile robots. In: *Robots and Sensor Clouds*. [S.l.]: Springer, 2016. p. 33–52. 41, 53 Citado 2 vezes nas páginas 41 and 53.

SANTOS, A. C.; TARRATACA, L.; CARDOSO, J. M. The feasibility of navigation algorithms on smartphones using j2me. *Mobile Networks and Applications*, Springer, v. 15, n. 6, p. 819–830, 2010. 17, 21 Citado 2 vezes nas páginas 17 and 21.

SAVALL, J.; AVELLO, A.; BRIONES, L. Two compact robots for remote inspection of hazardous areas in nuclear power plants. In: IEEE. *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. [S.l.], 1999. v. 3, p. 1993–1998. 13 Citado na página 13.

SCHILLER, J. H. *Mobile communications*. [S.l.]: Pearson Education, 2003. 23, 24, 25, 26 Citado 4 vezes nas páginas 23, 24, 25, and 26.

SHAH, D. *Gossip algorithms*. [S.l.]: Now Publishers Inc, 2009. 44 Citado na página 44.

SHINDE, S.; BORLE, V.; LONGJAM, A. Abhikaha: Aerial collision avoidance in quadcopter using cloud robotics. 2016. 41 Citado na página 41.

TAYLOR, R. H. et al. *Remote center-of-motion robot for surgery*. [S.l.]: Google Patents, 1995. US Patent 5,397,323. 15 Citado na página 15.

THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics*. [S.l.]: MIT press, 2005. 13 Citado na página 13.

TUTEJA, A.; GUJRAL, R.; THALIA, S. Comparative performance analysis of dsdv, aodv and dsr routing protocols in manet using ns2. In: IEEE. *Advances in Computer Engineering (ACE), 2010 International Conference on*. [S.l.], 2010. p. 330–333. 29 Citado na página 29.

VANDENBERGHE, W.; MOERMAN, I.; DEMEESTER, P. Adoption of vehicular ad hoc networking protocols by networked robots. *Wireless Personal Communications*, Springer, v. 64, n. 3, p. 489–522, 2012. 31, 32 Citado 2 vezes nas páginas 31 and 32.

WHITCOMB, L. L. Underwater robotics: Out of the research laboratory and into the field. In: IEEE. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. [S.l.], 2000. v. 1, p. 709–716. 13 Citado na página 13.

WHITE, T. *Hadoop: The definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2012. 43 Citado na página 43.

WU, X. et al. Proactive or reactive routing: A unified analytical framework in manets. In: IEEE. *2008 Proceedings of 17th International Conference on Computer Communications and Networks*. [S.l.], 2008. p. 1–7. 29 Citado na página 29.

YICK, J.; MUKHERJEE, B.; GHOSAL, D. Wireless sensor network survey. *Computer networks*, Elsevier, v. 52, n. 12, p. 2292–2330, 2008. 45 Citado na página 45.