Pós-Graduação em Ciência da Computação

Jairo Matheus Vilaça Alves

Um mecanismo de tomada de decisão de *offloading* de processamento em um cenário industrial

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2018

Jairo Matheus Vilaça Alves

Um mecanismo de tomada de decisão de *offloading* de processamento em um cenário industrial

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Computação em Nuvem
Orientador: Djamel Fawzi Hadj Sadok
Coorientador: Rafael Roque Aschoff

Recife
2018

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

A474m Alves, Jairo Matheus Vilaça
Um mecanismo de tomada de decisão de *offloading* de processamento em um cenário industrial / Jairo Matheus Vilaça Alves. – 2018.
55 f.: il., fig., tab.

Orientador: Djamel Fawzi Hadj Sadok.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências.

1. Ciência da computação. 2. Computação em nuvem. I. Sadok, Djamel Fawzi Hadj (orientador). II. Título.

004 CDD (23. ed.) UFPE- MEI 2019-020

Dissertação de Mestrado apresentada por **Jairo Matheus Vilaça Alves** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Um mecanismo de tomada de decisão de *offloading* de processamento em um cenário industrial**" **Orientador: Djamel Fawzi Hadj Sadok** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Djamel Fawzi Hadj Sadok
Centro de Informática/ UFPE

Profa. Glauco Estácio Gonçalves
Departamento de Informática / UFRPE

Profa. Arthur de Castro Callado
Departamento de Computação / UFC

Visto e permitida a impressão.
Recife, 10 de Setembro de 2018.

AGRADECIMENTOS

À Deus, por permitir que eu concluísse mais esta importante etapa da minha vida profissional.

Ao meu orientador, Professor Dr. Djamel Sadok, e coorientador Professor Dr. Rafael Aschoff, pelas valiosas sugestões, e pela confiança e estímulo na condução deste trabalho, mesmo nos momentos mais complicados.

Aos professores Dr. Arthur Calado e Dr. Glauco Gonçalves por terem aceito o convite em participar da minha banca e pelas excelentes sugestões para melhoria do trabalho.

Ao apoio do Grupo de Pesquisa de Redes e Telecomunicações da UFPE que por várias vezes demonstrou o seu comprometimento com o desenvolvimento de seus membros, incentivando o crescimento da ciência e da produção científica nacional, em especial aos Professores Dr. Djamel Sadok e Dr.ª Judith Kelner, pelo apoio e confiança nos momentos difíceis no grupo de pesquisa.

À minha mãe, Edilene, e irmã, Maíra, por sempre terem me encorajado aos estudos, não permitindo desistir dos meus objetivos e apoiando os meus projetos pessoais durante várias etapas da minha vida.

À minha namorada Júlia, pela paciência e compreensão durante o tempo ausente na fase de elaboração deste trabalho.

À Professora Dr.ª Patricia Endo, pela mentoria durante minha graduação e no início do mestrado e pela a indicação ao grupo de pesquisa.

À Wesley Davison, pelas incontáveis conversas, troca de experiências e conselhos profissionais e pessoais.

Por último quero agradecer aos amigos que fiz no grupo de pesquisa, Daniel Bezerra, Bruno Rodrigues, Silvia Ito, Késsia, Marcos Machado, Thiago Rodrigues, Gregório, Carolina Cani, Rafael, Demis, Guto, Gibson e Pedro, por tornar meus dias melhores, pelo apoio incondicional e pela ajuda nos momentos de indecisão.

RESUMO

Introduzida pela primeira vez na Alemanha, a indústria 4.0 engloba tecnologias utilizadas na automação de processos industriais. Algumas dessas são as Internet das Coisas, Inteligência Artificial e Robótica em Nuvem, que permitem o funcionamento de uma fábrica inteligente. Indústria 4.0 melhora os processos de produção, logística e serviço, e possibilita a introdução de métodos de produção eficientes e serviços industriais personalizados. Entretanto, existem alguns desafios de desempenho e energia a serem enfrentados para que as fábricas inteligentes se tornarem de fato eficientes e eficazes em seus processos. A Robótica em Nuvem permite a utilização de serviços remotos com a finalidade de ampliar os recursos computacionais dos robôs, mas traz consigo os mesmos desafios das fábricas inteligentes. Decisões como qual o local mais adequado para execução de processos de uma fábrica inteligente ou para qual recurso devem ser enviados os processos em caso de *offloading* de processamento, são o foco deste trabalho. A partir disso, foi proposto nesta dissertação um algoritmo de tomada de decisão de *offloading* de processamento de aplicações industriais, levando em consideração as diferentes capacidades dos recursos na nuvem e o tipo de rede (3G, 4G e *WiFi*). O cenário de fábrica foi simulado utilizando o Gazebo e as máquinas virtuais foram utilizadas como recursos de nuvem. O algoritmo proposto foi comparado com o algoritmo MinED já bem difundido na literatura. Os resultados obtidos comprovam a superioridade do algoritmo proposto para todos os tipos de rede considerados.

Palavras-chaves: Indústria 4.0. Robótica. Computação em Nuvem. Tomada de Decisão. Offloading.

ABSTRACT

Introduced for the first time in Germany, Industry 4.0 includes technologies used in the automation of industrial processes. Some of these are the Internet of Things, Artificial Intelligence and Cloud Robotics, which allow the operation of a smart factories. Industry 4.0 improves production, logistics and service processes, and enables efficient production methods and personalized industrial services. However, there are some performance and energy challenges to be faced in order to smart factories to become efficient and effective in their processes. Cloud Robotics allows the use of remote services to extend the computational capabilities of robots, but brings with it the same challenges as intelligent factories. Decisions such as where the most appropriate place to run processes from a smart factories or to which resource the processes should be sent in case of offloading processing, are the focus of this work. From this, a decision-making algorithm for offloading industrial applications was proposed, taking into account the different capacities of the resources in the cloud and the type of network (3G, 4G and WiFi). The factory scenario was simulated using the Gazebo and the virtual machines were used as cloud features. The proposed algorithm was compared with another algorithm already well known in the literature. The results obtained prove the superiority of the proposed algorithm for all network types considered.

Key-words: Industry 4.0. Robotics. Cloud Computing. Making Decision. Offloading.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Na última década, vimos o início do uso bem-sucedido e em larga escala de robôs móveis. No entanto, a maioria destes robôs continua a usar estratégias de controle simples (por exemplo, aspiradores robóticos), que demandam pouco processamento ou é operada remotamente por humanos (por exemplo, *drones*, veículos terrestres não tripulados e robôs de tele presença). Um motivo pela ausência de inteligência nos robôs móveis diz respeito aos custos elevados de computação e de armazenamento, afetando não só o preço do robô como também resultam na necessidade de espaço adicional para memória. Isto implicaria em peso extra para o robô, restrigindo a mobilidade e o tempo de operação do robô. Outro motivo é a ausência de mecanismo e ambiente de comunicação para compartilhar conhecimento entre robôs, principalmente nos casos com componentes de hardware e software diferentes (MOHANARAJAH et al., 2015).

Por outro lado, o rápido progresso da tecnologia redes sem fio e a alta disponibilidade de *data centers* oferecem aos robôs a possibilidade de acesso à nuvem. O uso da *Web* como um recurso computacional, meio de comunicação e fonte de informações compartilhadas pode permitir que os desenvolvedores superem as atuais limitações, construindo aplicações robustas de robótica (MOHANARAJAH et al., 2015). Exemplos destas aplicações incluem construção de mapas (ARUMUGAM et al., 2010), planejamento de tarefas (KEHOE; BERENSON; GOLDBERG, 2012), reconhecimento de objetos, localização e muitas outras. As aplicações de robótica em nuvem possuem o potencial para proporcionar robôs mais leves, por não necessitar de mais memória por exemplo, mais inteligentes, por utilizar recursos de nuvem que permite os robôs aumentarem sua capacidade de processamento, e mais rentáveis, por não necessitar de processadores mais robustos.

Em meio a tudo isso, foi introduzido na Alemanha o conceito de Indústria 4.0 (BRÜNGLINGHAUS, 2015), também chamada de "fábrica inteligente", no qual os sistemas ciberfísicos monitoram os processos físicos da fábrica e tomam decisões descentralizadas. Os sistemas físicos comportam-se como Internet das Coisas (do inglês *Internet of Things* - IoT), comunicando-se e cooperando entre si e com humanos em tempo real através da *Web* (FORBES, 2016). No contexto deste conceito, em 2010, James Kuffner introduziu o termo "*Cloud Robotics*"(KUFFNER, 2010) para descrever uma nova abordagem da robótica, que tem como objetivo o uso de tecnologias de nuvem com a finalidade de aumentar a capacidade de processamento e armazenamento dos robôs.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

De acordo com o relatório de 2017 da Federação Internacional de Robótica (FIR)(MINAMI, 2017), até 2020, estima-se que mais de 1,7 milhão de novos robôs industriais serão ins-

talados em fábricas por todo o mundo. Além disso, o relatório acrescenta que o total de vendas para todos os tipos de robôs domésticos (por exemplo, robôs aspiradores de pó, robôs cortadores de grama e robôs de limpeza de janelas) pode atingir quase 32 milhões de unidades no período 2018-2020, com um valor estimado em US$ 11,7 bilhões. Simultaneamente, estima-se que as vendas unitárias totais de robôs de serviço atinjam um montante aproximado de US$ 18,8 bilhões.

Além disso, um relatório do *Boston Consulting Group* (BCG) (BCG, 2015) projeta que o gasto do mercado global de robótica pode atingir US$ 67 bilhões até 2025. Nos últimos anos, a robótica assumiu novos papéis e se mostrou mais promissora do que o esperado.

Ainda segundo o BCG (BCG, 2015), dentre as maiores tendências que impulsionam a demanda encontram-se os veículos autônomos e as casas inteligentes. A queda dos preços também está tornando a robótica mais acessível. E isso está permitindo aos consumidores impulsionar a área de robótica, com empresas investindo em um futuro de automação, principalmente na área industrial (BCG, 2015). Na Figura 1, o setor indústrial foi o que mais cresceu até 2015 como mostra o estudo realizado pela BCG (BCG, 2015). E também segundo (BCG, 2015), é projetado que permaneça sendo a área de maior crescimento nos próximos anos até 2025.

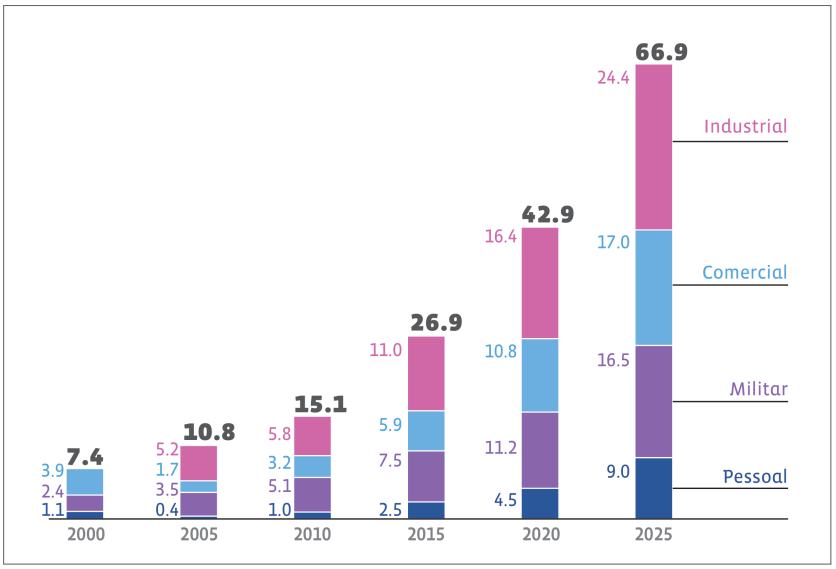

Figura 1 – Demanda por robótica. Adaptado de: Boston Consulting Group (BCG, 2015)

Neste relatório, a BCG também avaliou todo o espectro do mercado global de robótica, que compreende quatro setores:

- O segmento pessoal - robôs utilizados para entretenimento, limpeza, educação, segurança e aplicações domésticas - deve crescer mais rápido, a uma taxa anual composta de 15,8%, passando de US$ 1,0 bilhão em 2010 para US$ 9,0 bilhões em 2025.
- O segmento comercial - robôs utilizados para fins médicos e cirúrgicos, agricultura e construção, entre outras aplicações - deve crescer a uma taxa composta de 11,8% ao

ano, passando de US$ 3,2 bilhões em 2010 para US$ 17,0 bilhões em 2025, superando gastos com aplicações militares.

- O segmento militar - para veículos aéreos, terrestres e submarinos não tripulados, entre outras aplicações - deve crescer a uma taxa de 8,1% entre 2010 e 2025 e será o terceiro maior segmento, com US$ 16,5 bilhões.

- O segmento industrial - robôs utilizados em aplicações como soldagem, montagem, pintura e manuseio de materiais - continuará sendo o maior, crescendo a uma taxa de 10,1% ao ano, passando de US$ 5,8 bilhões em 2010 para US$ 24,4 bilhões em 2025.

Como pode ser observado, o segmento industrial é a área de robótica que mais poderá receber investimentos nos próximos anos. Tais investimentos justificam-se pelo fato do segmento em questão ser uma área crítica e complexa, crítica pois possui sistemas críticos, cuja falha ou mau funcionamento pode resultar em morte ou ferimentos graves em pessoas, ou perda ou dano severo aos equipamentos em volta, e complexa por possuir uma grande quantidade de processos sendo executados ao mesmo tempo e onde uma grande quantidade de dados deve ser transmitida de modo simultâneo. E com o advento da indústria 4.0, que introduzirá gradualmente sistemas autônomos nas fábricas, os robôs não serão apenas automatizados e sim mais inteligentes. Usando visão mecânica, sensores de movimento, reconhecimento de imagem e voz além de *software* sofisticados, os robôs serão capazes de lidar com trabalhos cada vez mais complexos, incluindo interagir e aprender continuamente com o ambiente e, especialmente, com os seres humanos.

No entanto, para que robôs possam realizar tudo o que foi citado, alguns desafios de computação necessitarão ser solucionados, por exemplo: o problema de baixa capacidade de processamento dos robôs, bem como a baixa capacidade de armazenar energia e do tempo de uso da bateria, aumentar a capacidade de armazenamento da memória, melhorar tempos de resposta entre o sensor e a atuação, pois os tempos de latência as vezes são alto, solucionar a variabilidade das conexões de rede devido os diferentes padrões de conectividade entre dispositivos diferentes.

Dentre esses problemas a serem solucionados, esta dissertação irá focar no problema de energia e baixa capacidade de processamento dos dispositivos móveis, considerando um cenário industrial, bem como nos tempos de resposta de baixa latência. Considerando as avaliações realizadas neste trabalho, os problemas da capacidade computacional e da latência podem ser tratados e resolvidos de uma única vez através da técnica de *offloading*, que permite descarregar o processamento das aplicações em um servidor remoto.

1.2 OBJETIVOS

O objetivo principal desta dissertação é propor um mecanismo que visa minimizar o *tradeoff* entre o desempenho e o consumo de energia para ambientes de robótica industrial. A metodologia é aplicada em um cenário de robótica industrial, com o objetivo de avaliar o impacto do *offloading* de processamento dos sistemas móveis em um ambiente industrial, utilizando a nuvem para melhorar o desempenho e a energia.

A seguir, apresentamos a lista de objetivos secundários deste trabalho:

- Construir uma aplicação de planejamento de movimentos de um braço robótico industrial;

- Propor um mecanismo de *offloading* que leve em consideração o tempo e a energia consumida na execução de uma aplicação industrial;

- Comparar o resultado do mecanismo de *offloading* proposto com outro mecanismo da literatura.

1.3 ESTRUTURA DA DISSERTAÇÃO

Este documento foi organizado da seguinte forma: o Capítulo 2, descreve um levantamento realizado de conceitos acerca dos temas: indústria 4.0, computação em nuvem, *Mobile Cloud Computing* (MCC) e robótica em nuvem (focando nas abordagens de tomada de decisão de *offloading*). O Capítulo 3, os trabalhos já realizados na comunidade científica foram introduzidos. O Capítulo 4, apresenta a proposta que norteia esta dissertação. O Capítulo 5, descreve o planejamento dos experimentos, visando validar a proposta e seus resultados. Por fim, o Capítulo 6 as conclusões e perspectivas futuras para para continuação deste trabalho são mostradas.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma visão geral sobre computação em nuvem, área que ganhou popularidade nos últimos anos. A computação em nuvem oferece processamento, armazenamento, serviços e aplicativos através da Internet. Além disso, a computação em nuvem facilita a redução do custo de capital, desacopla os serviços da tecnologia subjacente e oferece flexibilidade em termos de provisionamento de recursos. A definição e alguns tópicos tais como Computação Móvel em Nuvem e Robótica em Nuvem são introduzidas a seguir.

2.1 INDÚSTRIA 4.0

O termo Indústria 4.0 foi mencionado pela primeira vez na Feira de Hannover, Alemanha, com a apresentação da iniciativa *"Industry 4.0"* (BOSCH, 2012). Após a primeira Revolução Industrial (dita de "mecanização"), como um resultado da invenção da máquina a vapor, houve também um segundo movimento (dito "de produção em massa"), graças à ajuda da eletricidade, e também um terceiro (dito de "digitalização") que por sua vez é caracterizado pelo uso da Eletrônica e Tecnologia da Informação. O avanço destes movimentos marca o início da quarta Revolução Industrial, baseada no uso de *Cyber Physical Systems* (CPS) e da Internet das Coisas. A integração de tecnologias cibernéticas para habilitar produtos à Internet facilita a criação de serviços inovadores e obtém, entre outras coisas, diagnósticos, manutenções e operações baseados na Internet, de maneira econômica e eficiente (JAZDI, 2014). Além disso, a mesma integração ajuda na realização de novos modelos de negócios, conceitos operacionais e controles inteligentes, focando no usuário e em suas necessidades individuais. O objetivo da Indústria 4.0 é promover o surgimento de fábricas inteligentes, caracterizadas pelos seguintes pontos (WAHLSTER, 2012): redes inteligentes, mobilidade, flexibilidade e novos modelos de inovação.

Este trabalho foca no campo das redes inteligentes, que, com auxílio de tecnologias cibernéticas como serviços de comunicação sem fio e de telefonia fixa, atuadores e sensores inteligentes e tecnologias de telecomunicações, podem auxiliar na automatização de sistemas (JAZDI, 2014). Dentre as tecnologias capazes de auxiliar em tal automatização, estão a computação em nuvem e a *Mobile Cloud Computing*, que serão descritas nos próximos tópicos deste capítulo.

2.2 COMPUTAÇÃO EM NUVEM

A principal ideia por trás da computação em nuvem não é recente: na década de 1960, John McCarthy já imaginava que as instalações de computação seriam fornecidas ao público em geral como uma utilidade (DOUGLAS, 1966). O termo **nuvem** também foi

utilizado em vários contextos, como a descrição de redes *Asynchronous Transfer Mode* (ATM) na década de 1990. No entanto, foi somente depois que o CEO do Google, Eric Schmidt, usou a palavra para descrever o modelo de negócios de fornecimento de serviços na Internet em 2006, que o termo realmente começou a ganhar popularidade. Desde então, o termo computação em nuvem tem sido utilizado principalmente como marketing, em uma variedade de contextos e para representar diversas ideias distintas (ZHANG; CHENG; BOUTABA, 2010). No entanto, o *National Institute of Standards and Technology* (NIST) (MELL; GRANCE et al., 2011) definiu a computação em nuvem como:

"A computação em nuvem é um modelo para permitir acesso de rede conveniente e sob demanda a um conjunto compartilhado de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e lançados com o mínimo esforço de gerenciamento ou interação do provedor de serviços."

2.2.1 Tipos de Nuvem

Os modelos de implantação de computação em nuvem podem ser divididos em três categorias diferentes (ZHANG; CHENG; BOUTABA, 2010).

Em infraestrutura de **nuvem pública**, armazenamento e outras fontes são oferecidos aos usuários públicos por um provedor de serviços. Neste modelo, os aplicativos de processamento de dados são executados em fontes de uma infraestrutura criada por um provedor de serviços e arrendada pelo usuário. Esta solução é adequada especialmente para uso individual. A nuvem pública possui uma estrutura de baixa segurança em comparação com outras infraestruturas da nuvem, por isso esse tipo de nuvem oferece soluções de custo relativamente baixo e, geralmente, é tarifada com base no pagamento por uso. Pode inclusive ser oferecida gratuitamente a usuários individuais.

A infraestrutura de **nuvem privada** é criada e operada exclusivamente para uma única instituição / organização. O acesso público de terceiros não é permitido. Nesse tipo de infraestrutura, os recursos e equipamentos são armazenados internamente na organização ou através de um terceiro que represente a organização. A infraestrutura de nuvem privada é principalmente preferida por empresas e instituições que priorizam a privacidade dos dados. Embora o seu custo seja maior quando comparado com o da infraestrutura de nuvem pública, este modelo oferece vantagens atraentes em termos de processamento e armazenamento de dados *bakcup* e sobretudo segurança.

A infraestrutura em **nuvem híbrida** é uma composição de duas ou mais nuvens privadas, comunitárias ou públicas. Informações sensíveis e confidenciais ou aplicativos críticos são armazenados na nuvem privada interna à nuvem híbrida (parte da nuvem hibrida), enquanto as aplicações que requerem menos segurança são armazenadas na nuvem pública.

2.2.2 Modelos de Serviços

A computação em nuvem possui três modelos diferentes oferecendo três tipos de serviços principais (ZHANG; CHENG; BOUTABA, 2010): Infraestrutura como Serviço (do inglês *Infrastructure as a Service* - IaaS), Plataforma como Serviço (do inglês *Platform as a Service* - PaaS) e Software como Serviço (do inglês *Software as a Service* - SaaS), como pode ser visto na Figura 2. A seguir, serão descritos cada um destes modelos de serviço.

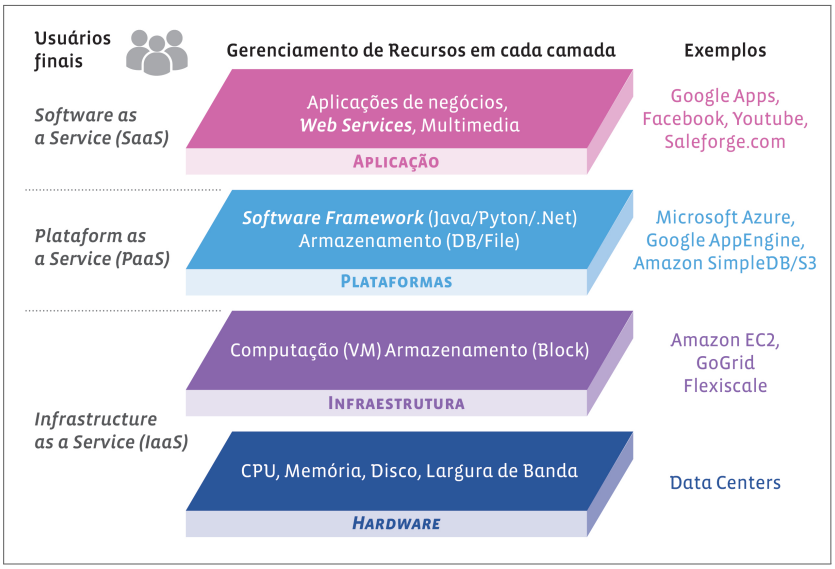

Figura 2 – Arquitetura de Computação em Nuvem. Adaptado de: Cloud computing: state-of-the-art and research challenges (ZHANG; CHENG; BOUTABA, 2010)

Software como um serviço (SaaS): os usuários podem acessar e interagir com as aplicações da nuvem através da Internet, usando interfaces como navegadores da web, sem a necessidade de instalar aplicativos em seus próprios sistemas (ZHANG; CHENG; BOUTABA, 2010). Em SaaS, o software é fornecido como um serviço através da Internet e o serviço é cotado tendo como base de pagamento o tempo de uso (ZHANG; CHENG; BOUTABA, 2010). No modelo SaaS, os usuários não gerenciam nem monitoram os componentes da infraestrutura, tais como rede, plataforma, sistema operacional e dispositivos de armazenamento. Os usuários só estão autorizados a alterar configurações específicas nas aplicações de acordo com o que foi contratado.

Plataforma como um serviço (PaaS): o provedor de serviços oferece aos usuários uma plataforma de computação para que possam desenvolver e executar suas próprias aplicações utilizando linguagens de programação, bancos de dados, serviços e ferramentas fornecidos pelo provedor (ZHANG; CHENG; BOUTABA, 2010). No modelo PaaS, os usuários não estão autorizados a controlar ou gerenciar servidores, sistemas operacionais, espaços de armazenamento e outros componentes que integram a infraestrutura da plataforma. O nível de autoridade dos usuários é limitado aos ajustes relacionados ao software transferido para a nuvem e às configurações da plataforma em que o software é executado.

Infraestrutura como um serviço (IaaS): no modelo IaaS, os usuários podem configurar processamento, armazenamento, redes e outros recursos de computação necessários para executar e instalar o sistema operacional e as aplicações necessárias (ZHANG; CHENG; BOUTABA, 2010). Os usuários não estão totalmente autorizados a gerenciar e controlar a infraestrutura física. No entanto, os usuários podem controlar o sistema no nível de armazenamento e sistema operacional e gerenciar componentes de rede específicos.

2.3 *MOBILE CLOUD COMPUTING*

Mobile Cloud Computing (MCC) pode ser definida como uma integração entre as tecnologias de computação em nuvem e dispositivos móveis, com a finalidade de oferecer a tais dispositivos recursos como poder computacional, memória, armazenamento e energia (SHIRAZ et al., 2013). MCC é o resultado de abordagens interdisciplinares que incluem computação móvel, computação em nuvem e redes sem fio. Portanto, esse domínio interdisciplinar também encontra-se referido como computação *mobicloud* (HUANG et al., 2011).

Da mesma forma que a MCC herda as características inerentes da computação em nuvem, como os tipos de nuvem e os modelos de serviço, ela herda também os principais problemas, como latência de rede e largura de banda em redes móveis, escalabilidade, segurança de dados e heterogeneidade dos ambientes móveis (GUAN et al., 2011). Quanto aos diferentes tipos de ambientes móveis (como telefonia móvel, redes veiculares e robótica móvel) esta dissertação aborda somente o ambiente de robótica em nuvem, que será descrito a seguir.

2.3.1 Robótica em Nuvem

Goldberg define a robótica em nuvem como (KEHOE et al., 2015): *"Qualquer robô ou sistema de automação que dependa de dados ou códigos de uma rede para suportar sua operação, ou seja, onde nem todos os sensores, processamento e memória estão integrados em um único sistema autônomo"*.

A Figura 3 ilustra o uso da robótica em nuvem, onde vários robôs de diferentes fabricantes e tecnologias diversas se conectam a uma única nuvem.

A robótica em nuvem é um campo emergente de robótica, introduzido com o advento da computação em nuvem, computação móvel e outras tecnologias da Internet, centradas em torno dos benefícios da infraestrutura convergente e dos serviços compartilhados. A robótica em nuvem permite que robôs se beneficiem dos recursos computacionais de armazenamento e comunicação de modernos *data centers*. Além disso, a robótica em nuvem diminui as despesas gerais para manutenção e atualizações reduzindo a dependência de *middleware* personalizado (WAIBEL et al., 2011).

Figura 3 – Figura ilustrativa do uso da Robótica em Nuvem. Adaptado de: A Survey of Research on Cloud Robotics and Automation (KEHOE et al., 2015)

A robótica em nuvem permite que os robôs aproveitem o rápido aumento das taxas de transferência de dados para descarregar tarefas que demandem uma melhor capacidade de processamento para a nuvem. Isso é interessante para os robôs móveis, cuja computação *on-board* implica em requisitos adicionais de energia que podem reduzir o tempo operacional restrigindo a mobilidade do robô, bem como aumentando os custos (WAIBEL et al., 2011).

2.3.2 *Robot Operating System*

O ROS é uma estrutura flexível para escrever software para robôs (ROS, 2018), desenvolvido pela *Open Source Robotics Foundation*. O ROS possui uma coleção de ferramentas, bibliotecas e convenções que visam simplificar a tarefa de criar um comportamento complexo e robusto para o robô em uma ampla variedade de plataformas robóticas (ROS, 2018).

O ROS possui uma arquitetura simples: a Figura 4 ilustra a comunicação entre os diferentes nós do ROS, facilitando a troca de informações entre módulos distintos dentro de uma mesma rede. No contexto desta rede, deverá sempre existir uma máquina conhecida como *master*. O *master* inicializará todos os serviços necessários para que os nós ROS se comuniquem entre si. Os nós ROS, por sua vez, podem estar sendo executados em uma mesma máquina ou em máquinas diferentes. A comunicação entre os nós é feita através do protocolo TCP/IP. Cada nó pode publicar e também se inscrever em tópicos. Esses tópicos são gerenciados pelo *master*. O tópico funciona como uma variável: em outras palavras, um nó pode tanto escrever valores nessa variável quanto ler os valores dessa variável. Dessa forma, diversos nós podem se comunicar uns com os outros.

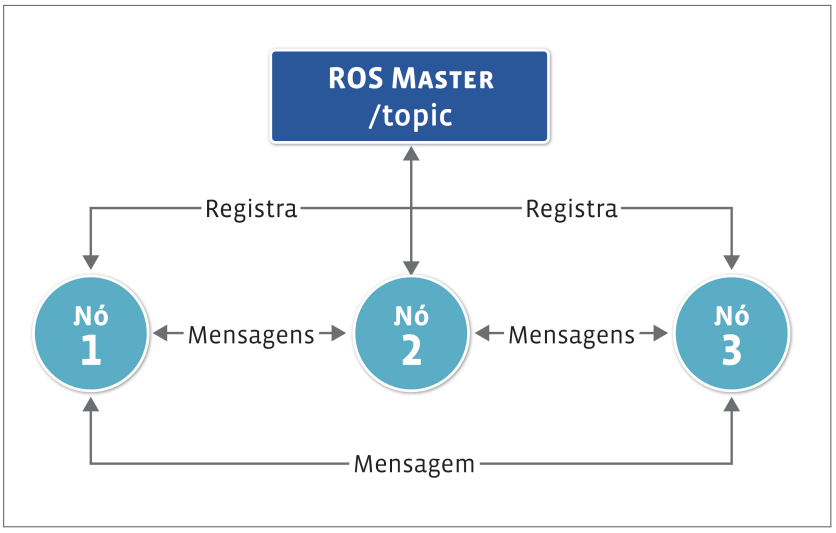

Figura 4 – Arquitetura do ROS. Adaptado de: Intro to ROS (ROBOTICS, 2015)

2.4 OFFLOADING

Offloading é um procedimento que migra processos intensivos e que demandam recursos de um disposito móvel para a nuvem, rica em recursos, proporcionada por um servidor remoto. Na Figura 5 podemos ver uma ilustração do que seria *offloading*, onde observamos a comunicação entre a nuvem e o braço robótico com o objetivo de trocr informações. O *offloading* de processamento baseado em nuvem melhora o desempenho das aplicações, reduz o consumo de energia da bateria e permite a execução de aplicações que de outra forma não poderiam ser executadas, devido aos recursos insuficientes de um dispositivo móvel. Além disso, a nuvem oferece serviços de armazenamento (AMAZON, 2017) que podem ser utilizados para superar as restrições de armazenamento dos dispositivos móveis. Atualmente, muitas aplicações existem com suporte de nuvem em diversos domínios, como comércio (YANG; PAN; SHEN, 2010), saúde (DOUKAS; PLIAKAS; MAGLOGIANNIS, 2010) (TANG; HU; HSU, 2010) e educação (FERZLI; KHALIFE, 2011) (ZHAO; SUN; DAI, 2010).

(KUMAR et al., 2013) analisa as abordagens mais comuns utilizadas na tomada de decisões sobre o *Offloading*, que são determinadas com base em vários fatores:

Porque fazer *offloading*, para melhorar o desempenho, reduzindo o tempo de execução no processamento de aplicações, e/ou melhorar a eficiência energética, reduzindo o consumo de energia.

Quando decidir fazer *offloading*, no momento de realização do *offloading*, o processamento pode se dar de duas formas: estático, onde o *offloading* do processamento será feito em algum dispositivo predeterminado, ou dinâmico, onde o *offloading* de processamento pode se dar em qualquer dispositivo que naquele momento tenha condições de executar mais rapidamente e com melhor eficiência energética. Em (KUMAR et al., 2013), é dito que, recentemente, o *offloading* dinâmico tem sido o mais escolhido pelos pesquisadores, devido a sua maior eficiência em comparação com o offloading estático.

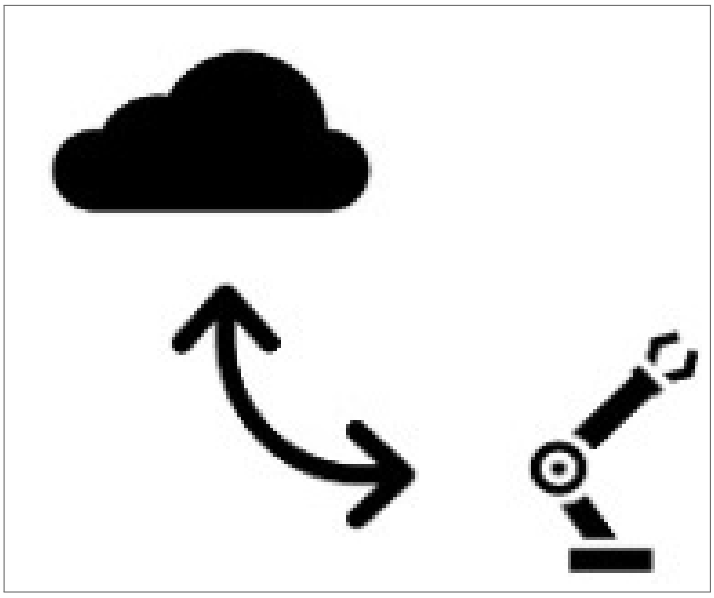

Figura 5 – Ilustração de *offloading*

Quais sistemas móveis usam o *offloading*, atualmente, várias pesquisas têm sido feitas para que sistemas móveis como carros, *smartphones*, robôs e sensores possam vir a fazer *offloading* de seus processos, aumentando assim as suas capacidades de processamento.

Tipos de aplicação, em (KUMAR et al., 2013), a pesquisa apresentada aponta aplicações que utilizam *offloading* de processamento, tais como: multimídia, jogos e cálculos matemáticos, por se tratarem de aplicações que demandam uma alta capacidade de processamento.

Infraestruturas para *offloading*, as principais estruturas para se fazer *offloading* atualmente são: computação em nuvem, *fog computing* e *mobile edge computing* (MEC). *Fog Computing* (BONOMI et al., 2014) (BONOMI et al., 2012) tem atraído interesse devido às vantagens propostas em relação às implementações centradas na nuvem, uma vez que mover grande parte do armazenamento/processamento para a borda da rede pode ser benéfico para ambientes de aplicações móveis. *Fog Computing* melhora o paradigma da computação de ponta a ponta, reduzindo problemas de contenção de recursos e escalabilidade (BONOMI et al., 2014). Alguns dos benefícios incluem: a) Redução de latência: reduzir os requisitos de transferência de dados pode ajudar a diminuir segundos cruciais na atuação dos dispositivos; e b) Largura de banda da rede: através da otimização dos dados que são transferidos via rede e das políticas necessárias associadas à borda (LUAN et al., 2015). *Mobile edge computing* possui os mesmo benefícios que *fog computing*, com a diferença de ser uma nuvem móvel (DOLUI; DATTA, 2017). A Figura 6 ilustra cada uma dessas infraestruturas. A subcamada contém cada dispositivos que precisam processar suas informações, bem como as tecnologias utilizadas para comunicação. Na camada intermediária estão algumas arquiteturas de *edge computing*, como *Fog Computing*, MEC

e *Cloudlets*. A *Fog Computing* apresenta uma camada de computação que aproveita dispositivos como *gateways* e roteadores sem fio. Esses dispositivos são usados para calcular e armazenar dados de dispositivos finais localmente antes de serem encaminhados para a nuvem (LUAN et al., 2015). Por outro lado, o MEC propõe a implementação de nós intermediários com capacidades de armazenamento e processamento nas estações base de redes celulares, oferecendo assim capacidades de computação em nuvem dentro da *Radio Area Network* (RAN) (LUAN et al., 2015). As *Cloudlets* são baseadas em dispositivos dedicados com capacidades semelhantes a um *data center*, mas em uma escala menor, presentes na vizinhança lógica dos consumidores. Esse paradigma permite que dispositivos finais descarreguem processamento para os dispositivos de *Cloudlet* com provisionamento de recursos semelhante ao de um *data center* (LUAN et al., 2015). Caso as aplicações dos dispositivos demandem um alta capadidade de processamento, pode ser utilizado a computação em nuvem na sobrecamada.

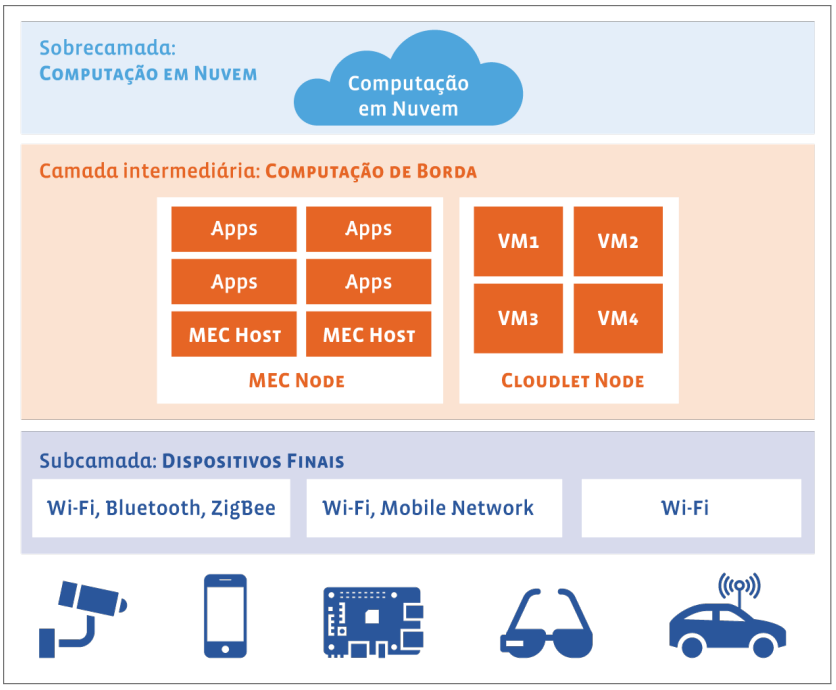

Figura 6 – Infraestruturas para *offloading*. Adaptado de (DOLUI; DATTA, 2017)

3 TRABALHOS RELACIONADOS

No estado da arte, diferentes esquemas de *offloading* foram propostos com o objetivo de otimizar o consumo de energia de sistemas móveis ou minimizar o custo do *offloading* das aplicações para nuvens remotas. Alguns trabalhos consideraram também a minimização de atraso de execução e, para isso, estes estudos determinam um prazo a fim de limitar o atraso na execução. Pois minimizar o próprio atraso de execução é particularmente importante para melhorar a qualidade dos serviços, especialmente para serviços on-line, como por exemplo, jogos de interação, pesquisas na *web* etc.

3.1 TRABALHOS ANALISADOS

Em (RAHMAN et al., 2016), foi desenvolvido pelos autores um framework de robótica em nuvem, que permite otimizar o desempenho das tarefas utilizando *offloading* para aplicações robóticas em cidades inteligentes que necessitam de qualidade de serviço (QoS). Os autores elaboraram dois modelos: um para eficiência energética e outro para atraso de execução. Para eficiência energética, foi utilizado o seguinte modelo:

$$E_{total} = \sum_{i=1}^{T} I_{t_i}.E_R\left(t_i\right) + \sum_{i=1}^{T} \neg I_{t_i}.E_C\left(t_i\right) \tag{3.1}$$

onde E_{total} é o consumo total de energia do robô, enquanto I_{t_i} denota as decisões de *offloading*, sendo 0 ou 1, e $\neg$ significa o operador NOT, quando I_{t_i} for 0, a energia do lado robô será 0, considerado o operador NOT em $\neg I_{t_i}$ a energia do lado da nuvem seria 1. Quanto a t_i, representa as tarefas, com i sendo o número de tarefas a serem executadas. Os custos de consumo de energia robótica para uma tarefa executada no robô e na nuvem são representados por E_R e E_C, respectivamente. Já para calcular o atraso de execução, os autores utilizaram o seguinte modelo:

$$T_{total} = \sum_{i=1}^{T} I_{t_i}.T_R\left(t_i\right) + \sum_{i=1}^{T} \neg I_{t_i}.T_C\left(t_i\right) \tag{3.2}$$

onde T_{total} representa o atraso total do robô enquanto I_{t_i} denota as decisões de *offloading* (que podem ser entre 0 ou 1) e $\neg$ como operador NOT. Custos de atrasos de execução de uma tarefa executada no robô e na nuvem são representados por T_R e T_C, respectivamente. Vale salientar que este trabalho não considerou a disponibilidade de recursos na rede, que é um importante fator a ser levado em consideração em caso de *offloading*. Outra questão diz respeito à última equação, que apenas utiliza o tempo de atraso de envio e recebimento de dados, deixando implícito outros fatores como oscilações da largura de banda, perda de pacotes de dados ou erros de transmissão. Como pode ser observado,

estes fatores não são tratados pela última equação, bem como não foram tratados pelo algoritmo utilizado no artigo.

(DEY; MUKHERJEE, 2016), por sua vez, elaboraram um esquema de *offloading* em que os robôs podem fazer processamento em um dispositivo *edge* ou na nuvem. Contudo, a nuvem só estaria conectada ao robô por meio destes dispositivos *edge*. O modelo proposto foi o seguinte:

$$T = min\left((x_1.E_r + x_2.W_e + x_3.W_c) + \rho.max\left(x_1.P_r + x_2.T_e + x_3.T_c\right)\right) \tag{3.3}$$

onde x_1, x_2, x_3 são a quantidade de processos atribuídos para o robô, *edge* e nuvem. E_r, W_e, W_c são os custos energéticos do robô, *edge* e nuvem para processar as tarefas. P_r, T_e, T_c são os atrasos de execução para o robô, *edge* e nuvem executarem as tarefas. Já ρ é um peso atribuído, embora seu valor não tenha sido informado pelos autores. T é apenas a métrica resultante da soma dos dois fatores considerados no estudo .O problema desse modelo encontra-se no dispositivo *edge*, que pode vir a ser um ponto de falha no sistema proposto. Caso o edge pare de funcionar, o robô fica sem opções para onde enviar seu processamento. Além disso, problemas de tráfego de rede ou confiabilidade podem levar a uma maior latência, o que poderia degradar o desempenho. Esta mesma situação também pode desperdiçar recursos disponíveis devido a problemas de conectividade e latência. Outro problema observado em algumas pesquisas (YOU et al., 2017) (WANG et al., 2018), é a inclusão de uma variável peso, que comumente tem valor desconhecido como no caso do artigo citado.

Os autores em (SEGATA et al., 2014) propuseram um modelo de *offloading* linear que mostra a razão entre consumo médio de energia por dados em *smartphones* para meios de transmissão distintos (2G, 3G e WiFi). Através de medições realizadas em diversos dispositivos, os autores chegaram à seguinte equação:

$$y\,[J] = \beta x\,[MB] \tag{3.4}$$

sendo $y\,[J]$ o consumo total de energia dado pelo produto do consumo médio de energia por Megabyte, β (simbolizando a relação de Joule por Megabyte), e a quantidade de dados transmitida, $x\,[MB]$. O fator β varia conforme a rede escolhida e o tipo de transmissão (*download* ou *upload*). Neste trabalho, o peso para fazer *offloading* de determinadas tarefas para a nuvem com economia de energia foi especificado pelos autores. Contudo, não foram realizadas análises quanto ao desempenho das aplicações no momento em que eram executadas na nuvem.

(MIETTINEN; NURMINEN, 2010) criaram um modelo para realização da tomada decisão de *offloading* baseado na seguinte inequação:

$$E_{envioERecebimentoNuvem} < E_{local}, \tag{3.5}$$

em que $E_{envioERecebimentoNuvem}$ representa o consumo de energia ao transmitir, executar e receber dados da nuvem e E_{local} representa a energia consumida no dispositivo móvel da aplicação executada localmente. No entanto, não foi feita qualquer análise sobre o desempenho quanto a tempo de execução das aplicações por parte dos autores, sob a justificativa de que a energia é o fator mais importante a ser considerado quando da realização do *offloading*.

(GUO et al., 2018), por sua vez, considera que existem três possíveis situações em uma decisão de *offloading* para um determinado local, sendo elas: (1) executar as aplicações localmente; (2) executar as aplicações em parte localmente e em parte na nuvem e (3) executar as aplicações totalmente na nuvem. Para cada uma das opções, foram elaborados três mecanismos a fim de avaliar o tempo de execução das aplicações. O problema desta abordagem, consiste na falta de um mecanismo adaptativo, que atendesse às três situações possíveis, pois da forma como foi abordado no artigo de (GUO et al., 2018), ele utiliza o mecanismo de cada vez em seus experimentos. Quanto à energia, os autores utilizam um mecanismo que contempla as três abordagens acima mencionadas, assumindo em seus cálculos que a energia consumida na nuvem é infinita, pois as máquinas/dispositivos na nuvem estão conectados as tomadas elétricas. Contudo, calcular a energia consumida pela aplicação na nuvem é importante, sobretudo em casos onde a nuvem é local. Assim, o custo total de energia de um local seria a soma dos resultados do processamento nas nuvens mais a soma dos resultados do processamento nos dispositivos que possuem tarefas a serem executadas.

Um dos trabalhos mais recentes sobre alocação de recursos na nuvem é o trabalho de (WANG et al., 2017). Os autores desenvolveram o esquema MinED visando minimizar o consumo de energia e o atraso no processamento das aplicações:

$$min \sum_{i=1}^{n} \sum_{j=1}^{A} \sum_{k=1}^{S} x_{i,j,k} d_{i,j,k} + \boldsymbol{a} \left(\sum_{i=1}^{n} \sum_{j=1}^{A} \left(1 - \sum_{k=1}^{S} x_{i,j,k} \right) e_{i,j} + \sum_{i=1}^{n} \sum_{j=1}^{A} \sum_{k=1}^{S} x_{i,j,k} e'_{i,j,k} \right) \quad (3.6)$$

onde n representa o número de dispositivos no sistema, A o número de aplicações e S o número de servidores. $x_{i,j,k}$ é booleano que denota a necessidade de realizar *offloading* da aplicação ou não, (se sim então a variável é 1, caso contrário é 0). i, j, k são respectivamente a quantidade de dispositivos, quantidade de aplicações sendo executada pelo dispositivo e quantidade de sevidores disponiveis para processamento das aplicações. $d_{i,j,k}$ é o atraso de enviar e receber os dados de uma aplicação em caso de *offloading*. $\boldsymbol{a}$ representa o peso, implicando em definir a prioridade entre energia e tempo de execução da aplicação. Caso a vida da bateria não consista num problema ou a aplicação seja sensível a atrasos, a prioridade será para o tempo e o peso atribuído para energia será configurado como 0. Caso contrário, o valor a ser atribuído será 1. $e_{i,j}$ representa o consumo de energia localmente e $e'_{i,j,k}$ o consumo de energia em caso de *offloading*. Este artigo se mostra eficiente para decidir onde deve ser feito o *offloading*, mas não considera quando fazer

offloading ou não, como em (KUMAR et al., 2013). É importante observar que, caso não seja necessário realizar o *offloading* e o $x_{i,j,k}$ for 0, o tempo de execução local será 0. Ou seja, o tempo para executar localmente não é levado em consideração, já que não é de fato calculado pela equação. Na Figura 7 temos a representação do mecanismo. Do lado esquerdo, temos o exemplo de processar uma aplicação remotamente, em que é calculado o consumo de energia do braço robótico e o atraso de execução remoto, que é o tempo de enviar e receber a troca de dados do servidor. Do lado direito, temos o processamento sendo feito todo no braço robótico e assim é calculado o consumo de energia de processar uma aplicação no braço robótico.

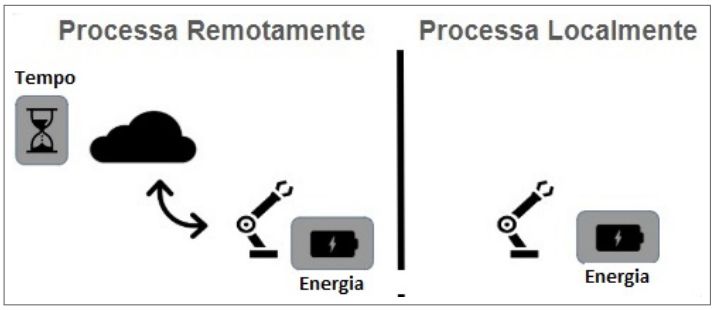

Figura 7 – Ilustração do *mecanismo MinED*

Em (YOU et al., 2017), (WANG et al., 2018) e (ZHANG et al., 2018), foram elaborados modelos matemáticos utilizando *Mobile Edge Computing* (MEC) com o objetivo de estender a capacidade dos dispositivos dos usuários. (YOU et al., 2017) utilizaram o sistema MEC para prolongar a vida útil da bateria e aumentar a capacidade de computação dos *smartphones*. Neste artigo, foi investigada a alocação de recursos para um sistema *Mobile-Edge Computing Offloading* (MECO) multiusuário baseado em acesso múltiplo por divisão de tempo (TDMA) e acesso múltiplo por divisão de frequência ortogonal (OFDMA). O algoritmo de alocação de recursos é formulado como um problema de otimização, para minimizar a soma ponderada do consumo de energia móvel sob a restrição de latência. Os resultados, segundo os autores, foram satisfatórios para minimizar o gasto de energia, ainda que os resultados não apresentam o ganho obtido.

(WANG et al., 2018), os autores consideram um sistema MEC multiusuário sem fio, onde usuários podiam executar suas respectivas tarefas localmente ou fazer *offloading* de todos ou de apenas uma parte para um *access point* (AP) com base em um protocolo TDMA. No entanto, os autores dos estudos (YOU et al., 2017) e (WANG et al., 2018) utilizaram pesos para determinar a prioridade de alguns usuários, mas tais pesos não foram explicitados em suas equações, o que impossibilita o uso destes modelos. (ZHANG et al., 2018) investigou o *trade-off* entre consumo de energia e o atraso de execução para o sistema MEC. Foi elaborado um esquema com dois *buffers*: um *buffer* que recebia as tarefas a serem processadas pelo servidor e outro *buffer* que guardava as tarefas que foram executadas pelo servidor. Depois de algum tempo, o dispositivo móvel baixaria os resultados do servidor.

O problema contido nesta abordagem é o acréscimo de um atraso desnecessário de espera nas tarefas no *buffer*.

3.2 DISCUSSÃO

Uma comparação de estratégias individuais de *offloading* de computação é ilustrada na Tabela 1. A maioria dos algoritmos de decisão de *offloading* visa minimizar o consumo de energia (E) enquanto o atraso de execução (T) for aceitável pela aplicação de *offloading* ou então visa analisar um *trade-off* entre essas duas métricas. Quanto ao tipo de *offloading* foi dividido em parcial e total, parcial no caso de o *offloading* ser dinâmico e total quando ele está sendo realizado *offloading* da aplicação toda para nuvem. Alguns artigos mais recentes utilizam MEC, despertando atenção significativa pelo desempenho para acelerar a operação das aplicações, enriquecendo a experiência do usuário. Além disso, todos os trabalhos validam as soluções propostas por simulação (apenas alguns artigos realizam avaliações analíticas ou análises teóricas).

Tabela 1 – Comparação entre os trabalhos

Trabalho	**Tipo de *Offloading***	**Objetivo**	**Método de Avaliação**
(RAHMAN et al., 2016)	Parcial	1) Minimizar E 2) Satisfazendo restrições de T	Simulação
(DEY; MUKHERJEE, 2016)	Total	1) Minimizar T 2) Satisfazendo restrições de E	Simulação
(SEGATA et al., 2014)	Total	1) Minimizar E	Ambiente Real
(MIETTINEN; NURMINEN, 2010)	Parcial	1) Minimizar E	Avaliação Analítica
(GUO et al., 2018)	Parcial e Total	1) Minimizar E 2) Satisfazendo restrições de T	Análise Teórica e Simulação
(WANG et al., 2017)	Parcial	1) Trade-off entre E e T	Análise Teórica e Simulação
(YOU et al., 2017)	Total	1) Minimizar E 2) Satisfazendo restrições de T	Simulação
(WANG et al., 2018)	Parcial	1) Minimizar E 2) Satisfazendo restrições de T	Simulação
(ZHANG et al., 2018)	Total	1) Trade-off entre E e T	Simulação

Como se pode perceber, várias pesquisas elaboraram também equações e inequações

visando avaliar o consumo de energia e o tempo de execução das aplicações. Por diversas vezes essas equações foram analisadas de maneira isolada, ou então as equações possuíam fatores importantes a considerar, mas que ficaram de fora das análises dos autores das pesquisas, como a disponibilidade de recursos para onde fazer *offloading*, bem como o tipo de rede. Questões de largura de banda e perda de pacotes foram minimizadas, sendo considerado apenas o tempo de envio e recebimento de dados.

Entretanto, a maioria dos autores limitou-se apenas a propor e validar modelos que visam decidir entre executar as aplicações localmente ou na nuvem. Poucos são os estudos recentes que elaboram uma estratégia de alocação de recursos para realização de *offloading* na nuvem, com objetivo de minimizar a energia e o tempo de execução das aplicações (no caso de as aplicações necessitarem ser executadas unicamente na nuvem, tendo em vista o crescimento do número de aplicações que necessitam de uma grande quantidade de processamento e que demandam máquinas mais robustas para seu processamento).

4 PROPOSTA

O principal objetivo desta dissertação de mestrado consiste na elaboração de um mecanismo de tomada de decisão de *offloading* que relaciona o desempenho, as conexões existentes (3G, 4G, *WiFi*) para execução remota e o tempo de execução em cada tipo de rede. Este capítulo apresenta o mecanismo proposto, a arquitetura da solução, o algoritmo e o cenário utilizados para realização da tomada de decisão de *offloading*.

4.1 CENÁRIO

Com objetivo de motivar pesquisas na área de robótica, foi desenvolvido uma competição chamada de *Agile Robotics for Industrial Automation Competition*(ARIAC), que é uma competição de robótica utilizando um cenário industrial que tem como objetivo testar a agilidade dos sistemas de robôs industriais, com a finalidade de permitir que os robôs industriais sejam mais produtivos, mais autônomos e mais sensíveis às necessidades de trabalhadores (ARIAC, 2017). Este cenário do ARIAC é utilizado nesta dissertação com a finalidade de executar o algoritmo de planejamento do braço robótico no cenário. O cenário é simulado na ferramenta *Gazebo*, que é capaz de simular com precisão e eficiência diversas ações de robôs em ambientes internos e externos complexos (GAZEBO, 2018), esse simulador foi desenvolvido utilizando a arquitetura *Robot Operating System* (ROS) explicada na Subseção 2.3.2.

Neste cenário é utilizado um sensor chamado de logical câmera que é responsável pela a localização dos objetos dentro do cenário. Objetos estes que serão utilizados para completar os objetivos do cenário. Outros sensores são utilizados, como de checagem de peças corretas e de presença de peças. O cenário possui também diferentes tipos de peças dispostas em mesas, como mostrado na Figura 8. Um braço robótico e dois *automated guided vehicle* (AGV). Os demais objetos são apenas para ilustrar um cenário de fábrica e nós não podíamos controla-los.

A competição possuía um único objetivo, que era montar um kit de peças de acordo com as ordens que eram dadas. Por exemplo, o cenário possui três tipos de peças (pistão, engrenagem e polia), uma ordem era recebida para que o robô montasse um kit de peças com 2 pistões, 2 engrenagens e 1 polia, então o braço robótico buscaria as peças nas mesas e colocaria no AGV, quando as peças chegavam no AGV, um sensor verificava se alguma peça era defeituosa, se a peça fosse defeituosa o braço robótica buscava outra peça e descartava a atual, caso não fosse, a ordem estaria completa.

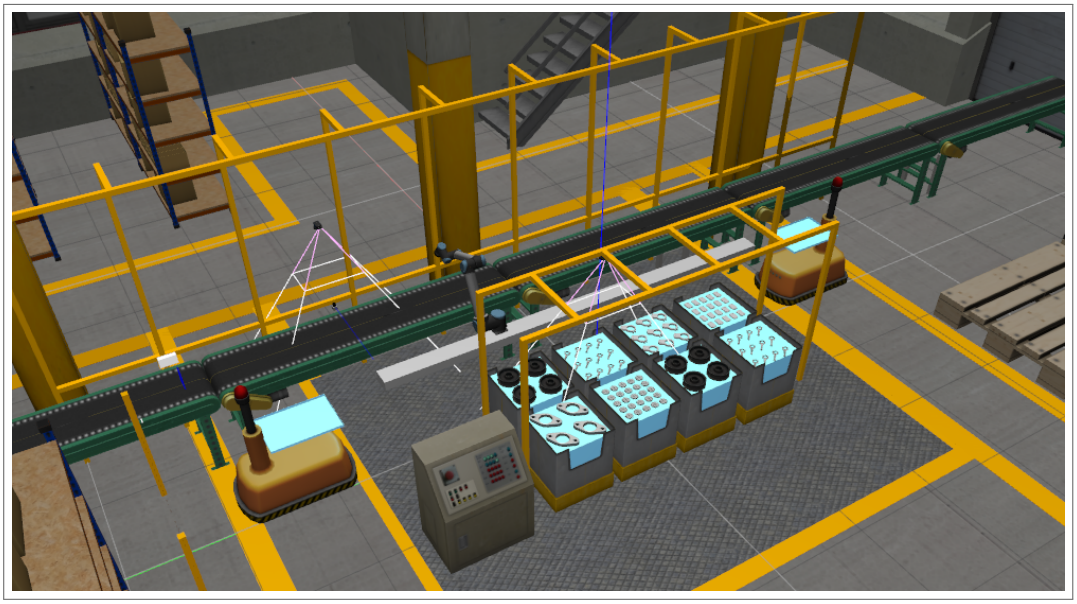

Figura 8 – Cenário do ARIAC. Imagem retirada de: ARIAC (ARIAC, 2017)

4.2 MECANISMO PROPOSTO

Nesta seção, será abordada a principal contribuição desta dissertação. A seguir iremos descrever o mecanismo proposto nesta dissertação, que tem como objetivo minimizar o *tradeoff* entre tempo e energia das aplicações que serão executadas no cenário do ARIAC, previamente descrito na Seção 4.1. Minimizar o gasto de energia e tempo é uma preocupação importante, sobretudo tendo em vista que os usuários desejam executar aplicações com baixo consumo de energia no menor tempo possível. Com base nesta necessidade e tendo como foco o cenário industrial do ARIAC, foi criado um mecanismo que leva em consideração o tempo e o consumo de energia do processamento das aplicações que estão sendo executadas por um braço robótico ou, em caso de *offloading*, pelos servidores.

4.2.1 Descrição do mecanismo

Nós consideramos $U = \{u_1, u_2, ..., u_i\}$ como o conjunto da quantidade de unidade móveis (como robôs) no cenário do ARIAC descrito na Seção 4.1, é possível denotar o conjunto de aplicações como $A = \{a_1, a_2, ..., a_j\}$, representando cada aplicação executada em uma unidade móvel como u_i. As aplicações executadas eram relativas ao planejamento das movimentações de uma braço robótico e de sensores espalhados pela fábrica. Assumindo que as aplicações a_j são executadas por k servidores, denotamos o conjunto dos servidores como $S = \{s_1, s_2, ..., s_k\}$. Contudo, nosso sistema possui uma única máquina cliente, que recebe os dados dos sensores e realiza o planejamento do robô, e portanto o valor da nossa unidade móvel U, sempre será 1.

Para cada aplicação a_j, se esta for executada localmente em uma unidade móvel, a energia consumida é $e_{i,j}$. Utilizamos $e'_{i,j,k}$ para denotar a energia consumida para fazer

offloading de uma aplicação de uma unidade móvel a_j para o servidor S_k. É importante ressaltar que a energia consumida nas unidades móveis ao executar localmente uma aplicação móvel é maior do que aquela utilizada para realizar *offloading* das aplicações móveis nos servidores em nuvem (WANG et al., 2017). Ainda para cada aplicação a_j, temos $t_{i,j,k}$ como o tempo de execução do processamento da aplicação feito localmente e $t'_{i,j,k}$ como o tempo de execução da aplicação em caso de *offloading* de processamento das aplicações para um servidor de nuvem S_k. Nesta variável, considera-se também o tempo de atraso para envio e recebimento no servidor do resultado da aplicação na unidade móvel.

Antes de formular o problema do mecanismo proposto, primeiro definimos a seguinte variável de decisão:

$$x_{i,j,k} = \begin{cases} 1, \text{com } \textit{ofloading} \\ 0, \text{sem } \textit{ofloading} \end{cases} \tag{4.1}$$

4.2.2 Formulação do mecanismo

Em primeiro lugar, para cada aplicação móvel, definida por a_j, caso seja executada localmente em uma unidade móvel, o tempo de execução será de:

$$\sum_{j=1}^{A_i} \left(1 - \sum_{k=1}^{S} x_{i,j,k}\right) t_{i,j} \tag{4.2}$$

Entretanto, se for realizado *offloading* a fim de executar em um servidor de nuvem representado como S_k, seu atraso de execução será de:

$$\sum_{j=1}^{A_i} \sum_{k=1}^{S} x_{i,j,k} t'_{i,j,k} \tag{4.3}$$

Então, o tempo total de execução das aplicações a_j pode ser escrito da seguinte forma:

$$\sum_{i=1}^{U} \sum_{j=1}^{A_i} \left(1 - \sum_{k=1}^{S} x_{i,j,k}\right) t_{i,j} + \sum_{i=1}^{U} \sum_{j=1}^{A_i} \sum_{k=1}^{S} x_{i,j,k} t'_{i,j,k} \tag{4.4}$$

Em segundo lugar, o consumo de energia de uma unidade móvel é dividido em duas partes: a energia usada para executar localmente algumas aplicações móveis a_j,

$$\sum_{j=1}^{A_i} \left(1 - \sum_{k=1}^{S} x_{i,j,k}\right) e_{i,j} \tag{4.5}$$

e a energia consumida para realizar *offloading* das aplicações móveis para os servidores em nuvem, que pode ser escrita da seguinte maneira:

$$\sum_{j=1}^{A_i} \sum_{k=1}^{S} x_{i,j,k} e'_{i,j,k} \tag{4.6}$$

Então, o consumo total de energia em um dispositivo móvel será dado por:

$$\sum_{i=1}^{U}\sum_{j=1}^{A_i}\left(1-\sum_{k=1}^{S}x_{i,j,k}\right)e_{i,j}+\sum_{i=1}^{U}\sum_{j=1}^{A_i}\sum_{k=1}^{S}x_{i,j,k}e'_{i,j,k} \tag{4.7}$$

Dessa forma, com o objetivo de minimizar o tempo de execução e o consumo de energia das aplicações a_j, é possível formular o mecanimo proposto como:

$$\begin{aligned} min\left(\sum_{i=1}^{U}\sum_{j=1}^{A_i}\left(1-\sum_{k=1}^{S}x_{i,j,k}\right)t_{i,j}+\sum_{i=1}^{U}\sum_{j=1}^{A_i}\sum_{k=1}^{S}x_{i,j,k}t'_{i,j,k}\right)+ \\ \boldsymbol{a}\left(\sum_{i=1}^{U}\sum_{j=1}^{A_i}\left(1-\sum_{k=1}^{S}x_{i,j,k}\right)e_{i,j}+\sum_{i=1}^{U}\sum_{j=1}^{A_i}\sum_{k=1}^{S}x_{i,j,k}e'_{i,j,k}\right) \end{aligned} \tag{4.8}$$

Geralmente, a depender do cenário onde a função acima será utilizada, os autores de (WANG et al., 2017) utilizam o parâmetro $\boldsymbol{a}$ para flexibilizar as preferências de *tradeoff*. Por exemplo, se a duração da bateria não for um problema grave ou a aplicação móvel for sensível a atraso, o parâmetro $\boldsymbol{a}$ pode ser definido como 0. Nesse caso, o problema torna-se apenas uma questão de minimizar o atraso, omitindo o consumo de energia. Como mencionado anteriormente, o objetivo do mecanismo é otimizar o tempo de execução e o consumo de energia entre processa localmente e remotamente, na Figura 9, temos a ilustração do mecanismo proposto. Do lado esquerdo, temos o exemplo de processar uma aplicação remotamente, em que é calculado o consumo de energia do braço robótico e o tempo de execução remoto, que é o tempo de enviar e receber a troca de dados do servidor e o tempo de processar a aplicação no servidor. Do lado direito, temos o processamento sendo feito todo no braço robótico e assim é calculado o tempo de execução do processo no braço, bem como o consumo de energia de processar uma aplicação no braço robótico. Essa proposta se difere da proposta do mecanismo MinED (WANG et al., 2017) pelo fato de eles não levarem em consideração o tempo de execução de processar a aplicação localmente, como pode ser visualizado na Figura 10.

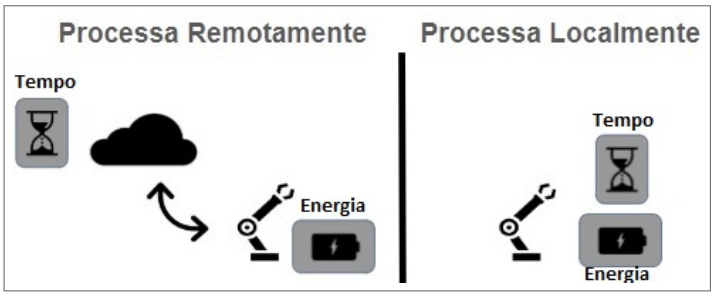

Figura 9 – Ilustração do *mecanimo proposto*

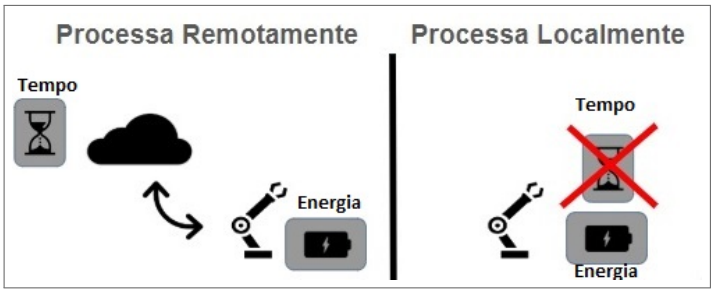

Figura 10 – Ilustração do *mecanismo MinED*

4.3 ARQUITETURA E DIAGRAMA DA PROPOSTA

Para esta competição, recebemos um ambiente simulado que representava um cenário industrial, tendo de desenvolver um algoritmo com a finalidade de cumprir as tarefas mencionadas na Seção 4.1. O sistema simulado era inteiramente baseado em ROS, descrito na Seção 2.3.2. A partir disso foi desenvolvido um algoritmo cuja arquitetura encontra-se representada na Figura 11. Esta mesma figura apresenta três componentes principais: *Gazebo Simulator*, tópicos do ROS e o *FIfth Generation MobilE Networks capabiliTies* (FIGMENT). Onde pude contribuir no componente FIGMENT com os módulos de *Arm Controller*, *Gripper Control* e *Inverse Kinematics Solutions*.

O componente do *Gazebo*, como mostrado na Figura 11, é dividido em 4 módulos. O módulo **Sensors Plugins** é responsável pelo funcionamento dos sensores dentro do cenário da competição, e utiliza sensores de proximidade para verificar a posição de uma peça a ser buscada na esteira rolante, checagem de defeitos para procurar falhas nas peças selecionadas pelo braço robótico e uma câmera, que informa a localização das peças em cima da mesa ao tópico do ROS *TF frames*. O módulo **DefaultRobotHWSim** é o módulo responsável pela física dos robôs inseridos no cenário do ARIAC. Foi adotado o braço robótico UR5 da *Universal Robots* para utilização no ambiente e também robôs AGV, que são robôs que recebem as peças e utilizam sensores para procurar por defeitos nas peças recebidas. O módulo **Gripper Plugin** tem como função posicionar a peça de acordo com a orientação correta no AGV, de acordo com as orientações *roll*, *pitch* e *yaw*, que são os ângulos das orientações fornecidas pelo módulo **Competition Management**. Este último módilo age como o "cérebro"da competição, sendo responsáve pela a inicialização e finalização da competição. O módulo também informa quais peças devem ser buscadas pelo braço mecânico e como o kit de peças deve ser montado, de acordo com a posição e orientação em que peça deveria ficar em cada um dos AGVs. Por último, o módulo também deve informar quais os tipos de peças que devem ser buscados no ambiente da competição.

Já o componente ROS *Topics/Services* é a camada pela qual o ROS é responsável. Este, possui 7 tópicos e 3 serviços. Cada tópico era atualizado pelo *Gazebo* e utilizado pelo FIGMENT, o tópico **TF Frames** recebe do *plugin* câmera as posições das peças em cima

das mesas. O módulo **Sensors Output** recebe os dados de todos os sensores do *Gazebo* como, por exemplo, ser informado neste tópico a cada vez que uma peça passava pelo sensor de proximidade na esteira rolante. O **Joint Trajectory Controller** é o tópico utilizado pelo *Arm Controller* do componente FIGMENT para realizar os movimentos das trajetórias a serem tomadas pelo braço robótico. O **Joint State Publisher** informa ao *Arm Controller* a posição atual das juntas do robô. O tópico **Orders** é gerenciado pelo *Competition Management*, sendo o tópico que contém a lista de ordens a serem executadas pelo braço robótico. O **AGV State** notifica sobre as peças contidas no AGV e o *Competition Management* notifica se estas peças foram colocadas de acordo com a posição e orientação corretas informadas pelo tópico *Orders*. O **Gripper State** recebe ordens do *Gripper Control* para pegar ou soltar uma peça e anuncia o estado atual (se a peça está no gripper ou não). Já os serviços, eles eram todos utilizados pelo o componente FIGMENT. Pelo serviço **Gripper Control**, o módulo *Gripper Control* do FIGMENT poderia ajustar as orientações das peças. O serviço AGV era avisado pelo módulo *Orders Manager* do FIGMENT, sinalizando que todas as peças foram entregues e a tarefa concluída. O **Competition** é o serviço utilizado para iniciar a competição, sendo utilizado *Planning* do FIGMENT.

O FIGMENT é o componente responsável por todo o planejamento das ações no cenário do ARIAC. O componente FIGMENT foi dividido em 6 módulos: o **Planning**, que controla todos os outros módulos do FIGMENT e inicializa e finaliza a competição quando todas as peças tiverem sido entregues, o **Arm Controller**, que controla as ações do braço robótico, o **Inverse Kinematics Solutions**, que faz os cálculos de cinemática inversa do robô, **Transforms Manager**, que gerencia o tópico *TF frames* (que permite localizar as peças a serem buscadas dentro do cenário), o **Orders Manager**, que recebe as ordens de qual peça deve ser pega, sendo essas ordens dadas pelo sistema do ARIAC e o **Gripper Control**, que é responsável pelas ações da garra do robô. Os módulos de *Arm Controller*, *Inverse Kinematics Solutions* e *Gripper Control* são os módulos desenvolvidos pelo autor deste trabalho para a competição do ARIAC, constituindo portanto em contribuições secundárias desta dissertação. Todos esses módulos são responsáveis pelo planejamento dos movimentos do robô. Primeiramente, o *Planning* inicia a competição, e as ordens sobre quais peças devem serem buscadas para montar o kit de peças são anunciadas no tópico *Orders* e identificadas pelo *Orders Manager*. Com isso, o *Planning* notifica o módulo *Transforms Manager* de que as ordens foram recebidas, e então este segundo módulo consulta o tópico *TF Frames* onde estão as peças a serem buscadas no cenário do ARIAC. Depois de saber onde estão as peças, o *Planning* define a movimentação que deve ser feita a fim de buscar a peça na mesa correta, a partir do *Inverse Kinematics Solutions*. Depois, utiliza o *Arm Controller* para controlar o braço e buscar as peças no local desejado. Chegando ao local, o *Planning* sinaliza para que o módulo *Gripper Control* pegue a peça.

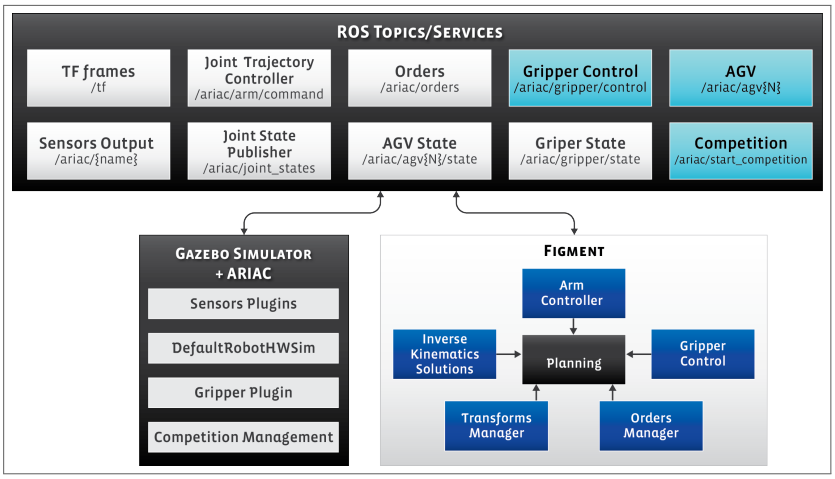

Figura 11 – Arquitetura da Competição. Fonte: autor.

Com essas ações de planejamento, a proposta deste trabalho pretende determinar se as ações executadas pelo *Planning* devem ser feitas em uma máquina local ou na nuvem. A Figura 12 ilustra como é feita a tomada de decisão de *offloading*. A máquina cliente executa a aplicação do *Gazebo*, representada pelo componente *Gazebo Simulator* na Figura 11. Quando essa aplicação é iniciada, o ROS também é inicializado, bem como todos os tópicos e serviços que serão utilizados pelos componentes *Gazebo Simulator* e FIGMENT da Figura 11. Na etatap 1 na Figura 12, é iniciado a aplicação, a partir disso é verificado a disponibilidade dos recursos de nuvem, e caso o servidor esteja funcionando, é coletado informações sobre o consumo energético e o tempo de execução do planejamento do componente FIGMENT na nuvem, e a partir deste tempo é feita a tomada de decisão da alocação de recursos na Etapa 2, decidindo qual máquina irá executar o planejamento do componente FIGMENT. Após os recursos serem alocados, o planejamento é executado, como pode ser visto na Etapa 3.

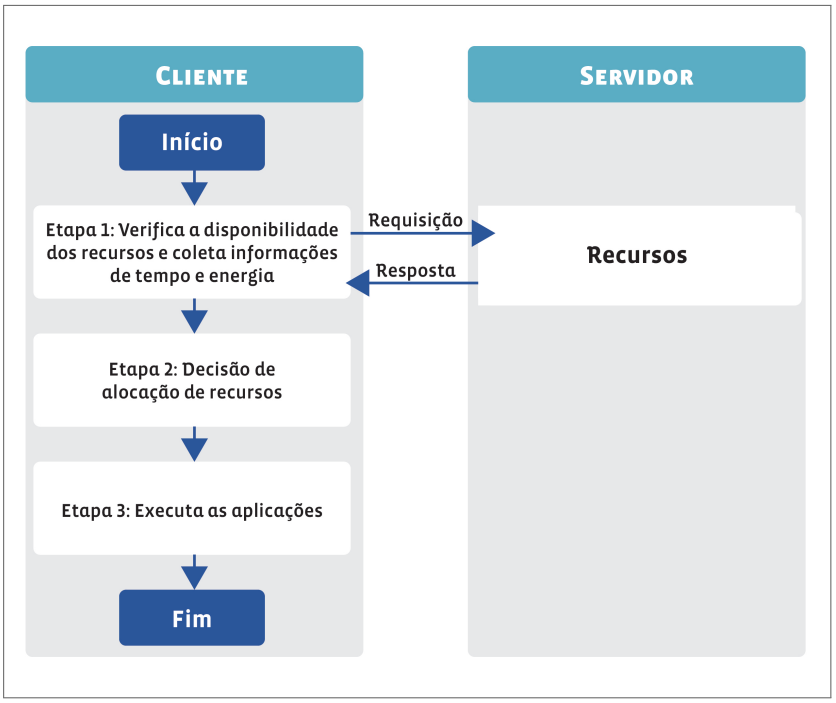

Figura 12 – Diagrama ilustrativo da tomada de decisão de *offloading*. Fonte: autor.

4.4 DIFICULDADES ENCONTRADAS

Inicialmente, foi feita uma tentativa de montar um cenário utilizando uma Raspberry, com o objetivo de realizar o processamento das ações do robô no simulador do *Gazebo*. A Raspberry possuía a versão 3, com processador *Broadcom* BCM2837 de 64 bits e clock de 1.2GHz. Essa abordagem foi pretendida levando em consideração que alguns sistemas de automação utilizam uma Raspberry para fazer processamento de suas aplicações, como em (YOUNIS et al., 2017) (RAJPUT et al., 2018), onde os autores das duas propostas desenvolveram um sistema de automação residencial baseado em fala. Então com o objetivo de utilizar um recurso com capacidade de processamento parecida com a de um robô, a Raspberry foi escolhida.

Contudo, um problema foi encontrado com relação à biblioteca do ROS durante a configuração do ambiente na Raspberry, pois uma das bibliotecas responsáveis por permitir a compilação do ROS apresentava erros. Ao pesquisar a solução do problema na comunidade ROS, percebeu-se que existe pouco suporte da comunidade na resolução de erros relacionados à Raspberry. Além disso, o erro encontrado durante a compilação do ROS não foi solucionado por nenhum membro da comunidade, sendo descoberto posteriormente que se tratava de uma pendência a ser solucionada pelos desenvolvedores do ROS para Raspberry. Outro fator que impediu o uso da Raspberry tinha relação com as bibliotecas utilizadas pela competição do ARIAC. O sistema operacional *Linux* da Raspberry não possui repositórios que permitam a instalação dos pacotes do ARIAC.

A partir do que foi exposto, e para contornar esses problemas, foi utilizado um compu-

tador *desktop* com o VMWare para simular e processar localmente a aplicação FIGMENT no ambiente do *Gazebo*. Com isso, após escolher onde seria executada a aplicação, fomos em busca de um *software* que pudesse medir o consumo de energia da CPU ao executar a aplicação. Algumas ferramentas de *software* foram investigadas e testadas e dentre todas as ferramentas, foi escolhida a *Powerstat*, por atender às necessidades deste estudo e por ser a mais utilizada de acordo com o que foi apurado nas comunidades *Linux*. Embora existam também algumas soluções em *hardware*, estas precisam ser instaladas fisicamente nos dispositivos em que se quer realizar a medição, o que acaba por contar à favor das ferramentas de *software*, de fácil instação. Contudo, a máquina virtual não permite a utilização do *software* de medição de energia e a partir disso, foi utlizado apenas a máquina *desktop* para realizar os experimentos.

5 VALIDAÇÃO DA PROPOSTA

Neste capítulo, é descrito o planejamento dos experimentos realizados (três ao todo), e é avaliado o desempenho do mecanismo proposto por meio de simulações. Para demonstrar a eficiência da tomada de decisão do mecanismo proposto, é comparado seu desempenho com a solução MinED, explicada no Capítulo 3.

5.1 PLANEJAMENTO DOS EXPERIMENTOS

No primeiro experimento, é medida a tomada de decisão sobre o *offloading* em uma rede 3G, comparando o mecanismo proposto de decisão de *offloading* com o mecanismo MinED (WANG et al., 2017), utilizando o tempo de execução, que é o tempo de enviar e receber o processamento da aplicação mais o tempo de processamento, e o consumo de energia como fatores. No entanto, é válido salientar que o mecanismo MinED não calcula o tempo de processamento no servidor, apenas o atraso de enviar e receber, mas para fins de comparação, foi consderado o tempo de processamento no servidor. No segundo experimento, são comparados os mesmos mecanismos, porém utilizando uma rede 4G. No terceiro experimento foi utilizada uma rede *Wifi*.

Para execução dos experimentos, foram utilizadas três máquinas *desktop*, sendo uma máquina local (ML) e duas máquinas remotas (servidores MS1 e MS2), como mostrado na Figura 13. As configurações de cada máquina podem ser vistas na Tabela 2.

A máquina local estava executando o simulador do *Gazebo*, simulando um cenário industrial e executando o algoritmo utilizado pela equipe do projeto FIGMENT para execução das aplicações dentro do ambiente simulado. Para emular o tipo de rede, foi utilizado o *Netem*, uma ferramenta específica para este propósito. O *Netem* permite controlar a latência, largura de banda, perda de pacotes, duplicação e reordenação de pacotes na rede (NETEM, 2016). A partir dessa ferramenta, foi possível alterar as medidas de latência da rede de acordo com as medidas da ferramenta *Network Performance* do Google (GOOGLE, 2018). A partir do *Network Performance*, são apresentados os valores da Tabela 3, que são os valores utlizados para emular o acesso à nuvem. Esses valores foram configurados de forma estática no emulador, considerando que a rede dentro de uma fábrica não sofre oscilações bruscas, como acontece, por exemplo, na rede de um *smartphone* em um carro em movimento.

O Experimento 1 consistiu em executar 30 vezes a aplicação de planejamento do robô com base nos seguintes fatores: tempo de processamento e consumo de energia. A comunicação entre o robô e a nuvem foi emulada considerando as redes 3G. Para os experimentos 2 e 3, executado também 30 vezes, foi alterado o tipo de rede, considerando respectivamente a rede 4G e *WiFi*. Estas informações encontram-se explicitadas nas Tabelas 4, 5 e

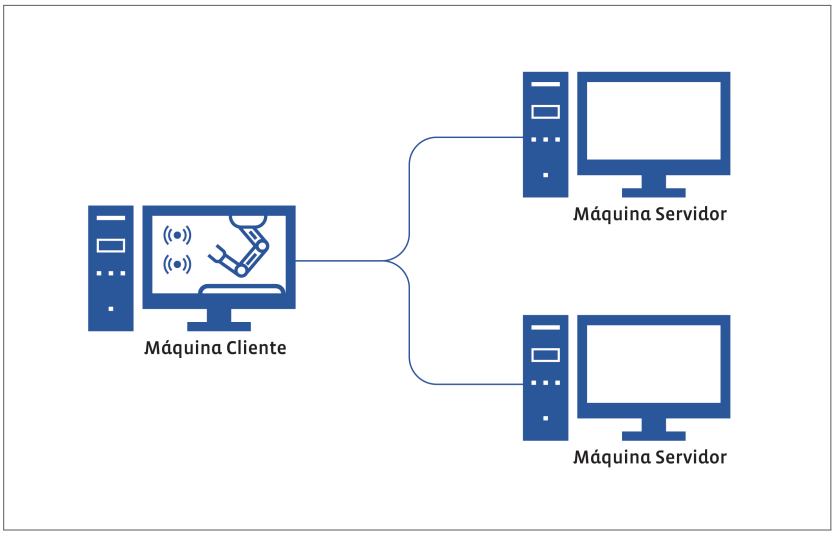

Figura 13 – Ambiente de Execução. Fonte: autor.

Tabela 2 – Ambiente de Testes

Máquina	Tipo	RAM	Processador	S.O	Velocidade(GHz)
desktop	Cliente	4 GB	Intel I3 (Geração 3)	Ubuntu 16.04	3.3 GHz (4 Core)
desktop	Servidor	8 GB	Intel I5(Geração 3)	Ubuntu 16.04	3.1 GHz (4 Core)
desktop	Servidor	8 GB	Intel I5(Geração 3)	Ubuntu 16.04	3.2 GHz (4 Core)

Tabela 3 – Parâmetros de Rede

Tipo de Rede	Latência
3G	100ms
4G	20ms
WiFi	2ms

6.

Tabela 4 – Experimento 1

Itens	**Descrição**
Fatores	Experimento 1: Tempo de execução e Consumo de Energia
Carga de Trabalho	Experimento 1: Processamento do planejamento do robô
Técnica de Análise	Simulação no Gazebo
Tipo de Rede	Experimento 1: 3G
Tipo de Offloading	Experimento 1: Estático

A coleta dos dados de energia foi realizada através da ferramenta *Powerstat* (HECTIC, 2012), que é capaz de estimar o consumo energético dos processos de sistemas operacionais. A ferramenta foi escolhida por permirtir à realização de estimativas baseadas em consumo energético unicamente de uma determinada aplicação (diferentemente de soluções em *hardware*, que realizam a captação de dados ligados ao consumo geral do sistema).

Tabela 5 – Experimento 2

Itens	**Descrição**
Fatores	Experimento 2: Tempo de execução e Consumo de Energia
Carga de Trabalho	Experimento 2: Processamento do planejamento do robô
Técnica de Análise	Simulação no Gazebo
Tipo de Rede	Experimento 2: 4G
Tipo de Offloading	Experimento 2: Estático

Tabela 6 – Experimento 3

Itens	**Descrição**
Fatores	Experimento 3: Tempo de execução e Consumo de Energia
Carga de Trabalho	Experimento 3: Processamento do planejamento do robô
Técnica de Análise	Simulação no Gazebo
Tipo de Rede	Experimento 3: *WiFi*
Tipo de Offloading	Experimento 3: Estático

Portanto, assumiu-se neste trabalho que tal ferramenta possui uma estimativa razoável para o estudo aqui realizado.

5.2 RESULTADO DOS EXPERIMENTOS

Os resultados dos experimentos das Figuras 14 e 15 representam o tempo de execução do planejamento do robô para os tempos de latência de 2ms, 20ms e 100ms, como mostrado na Tabela 3. Como esperado, a soma do tempo de ida e volta dos dados e do processamento de planejamento do robô, é maior para quando a latência é maior. Isto ocorre porque a aplicação de planejamento necessita de informações dos tópicos do ROS, que por sua vez demoram para atualizar quando a latência é maior. Assumindo que a distribuição dos resultados das Figuras 14 e 15 segue uma normal para o intervalo de confiança de 95%, são apresentadas as seguintes Figuras 16 e 17, que representam a média dos dados e o intervalo de confiança. Por estas figuras, é possível notar que a diferença entre as médias para as latências 2ms e 20ms é menor do que para as latências 20ms e 100ms. Isso implica dizer que, quanto maior a latência, pior o tempo de execução.

Nas Figuras 18 e 19, pode-se ver o consumo de energia na máquina local dos experimentos quando a aplicação de planejamento é executada nas máquinas servidoras MS1 e MS2, considerando as latências 2ms, 20ms e 100ms. O consumo de energia sempre esteve entre o intervalo de 6 e 8 Watts, com exceção das rodadas de execução 23 e 24 na Figura 18 e 14 na Figura 19. O consumo de energia mostrado nos gráficos, é o consumo de energia médio do tempo de processamento para cada rodada de execução. Por exemplo, na Figura 14, na primeira rodada de execução, o tempo de processamento do planejamento

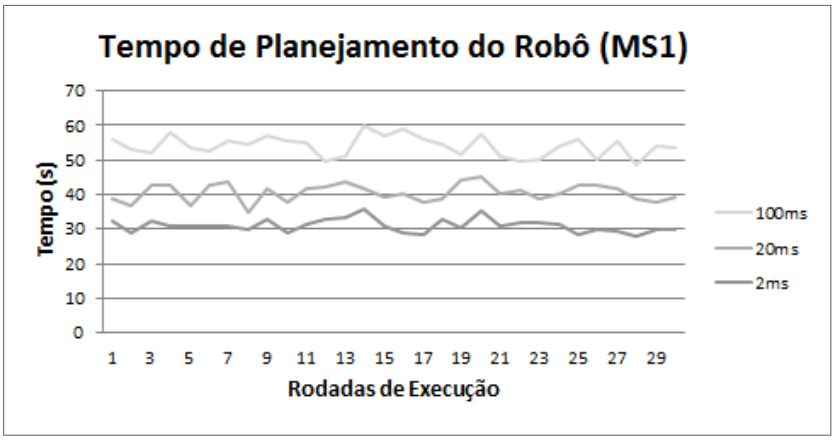

Figura 14 – Tempo de Planejamento em MS1

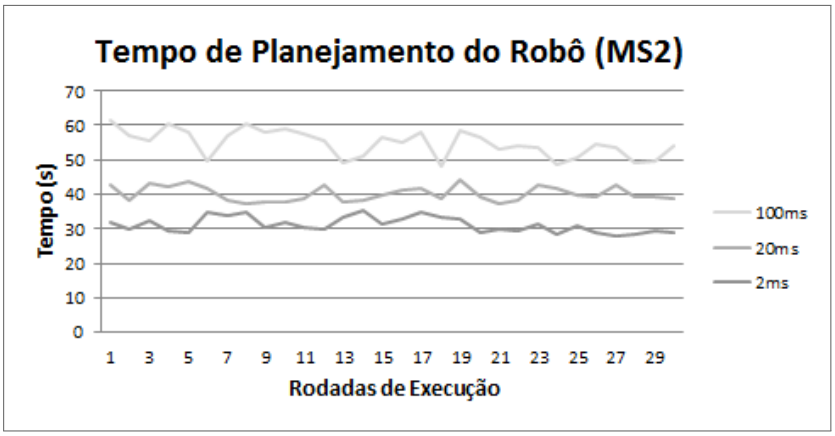

Figura 15 – Tempo de Planejamento em MS2

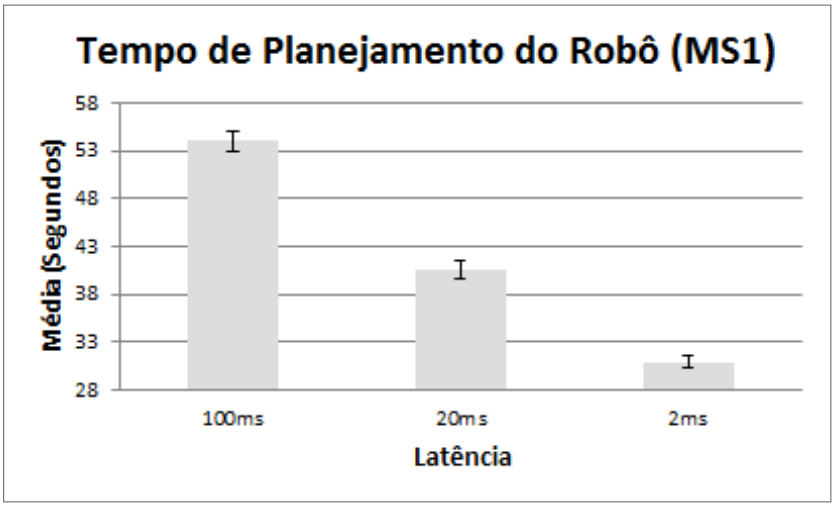

Figura 16 – Tempo Médio de Planejamento em MS1

do robô quando a latência foi de 100ms, foi de 56s e durante esse tempo, foi somado o consumo de energia de cada segundo do processamento da aplicação e calculado depois o consumo médio por segundo, que é o resultado mostrado na Figura 18. Assumindo que os resultados das Figuras 18 e 19 segue uma distribuição normal, com intervalo de confiança

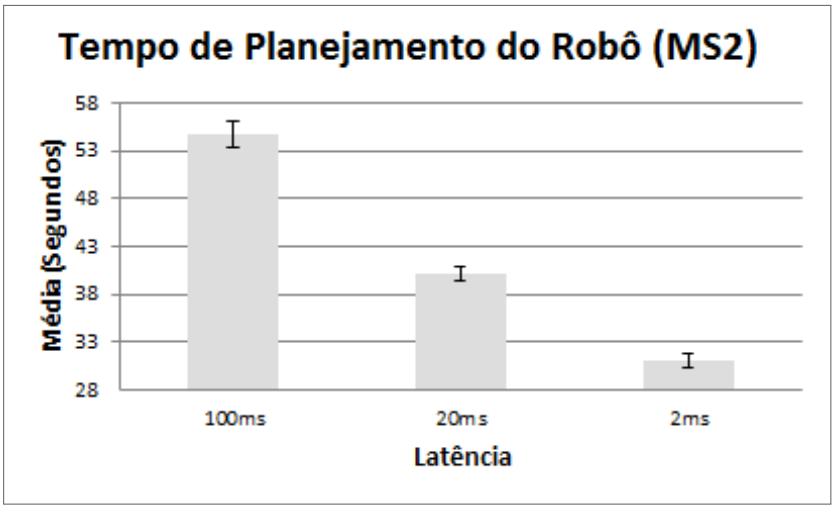

Figura 17 – Tempo Médio de Planejamento em MS2

de 95%, é possível obter como resultado as Figuras 20 e 21, que representam o consumo médio de energia para todas as latências e o intervalo de confiança. Como é observado nestas figuras, o consumo de energia para as diferentes latência são bem próximos uns dos outros e praticamente não se diferem.

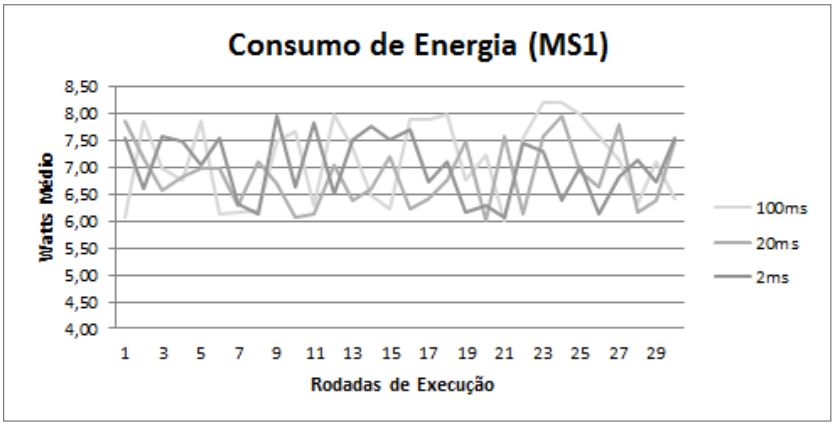

Figura 18 – Consumo de Energia em MS1

Para comparar os resultados da execução do planejamento do robô nos servidores MS1 e MS2 e em uma máquina local, é obtido a Figura 22. Quanto ao tempo de execução, a máquina local obteve o melhor tempo de processamento para as rodadas de execução 2,3,8,12,13,14, 16, 22, 23 e 24, quando levado em consideração o tempo de latência de 2ms. A partir disso, ao considerar apenas o tempo de execução como tomada de decisão, na maioria dos casos seria necessário fazer *offloading* do processamento. Já para o tempo de latência de 20ms, é obtido a seguinte Figura 23. Em todas as rodadas de execução, o tempo de processamento foi melhor ao ser executado na máquina local, devido ao aumento da latência. Na Figura 24, é mostrado um tempo de latência de 100ms e pode-se observar que os tempos de execução para execução nas máquinas servidores MS1 e MS2, aumenta significativamente. E em todas as rodadas de execução o tempo na máquina local foi

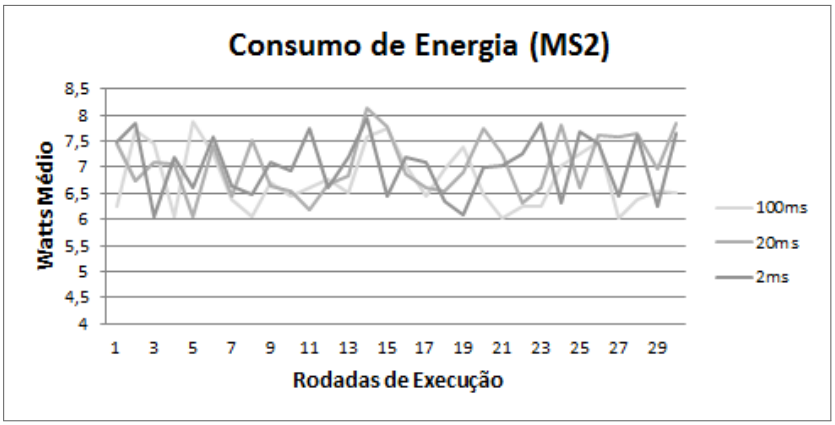

Figura 19 – Consumo de Energia em MS2

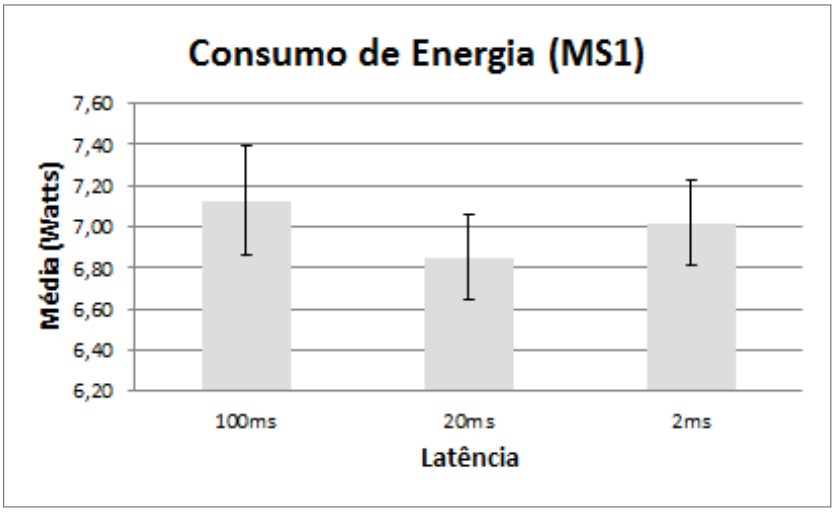

Figura 20 – Consumo Médio de Energia em MS1

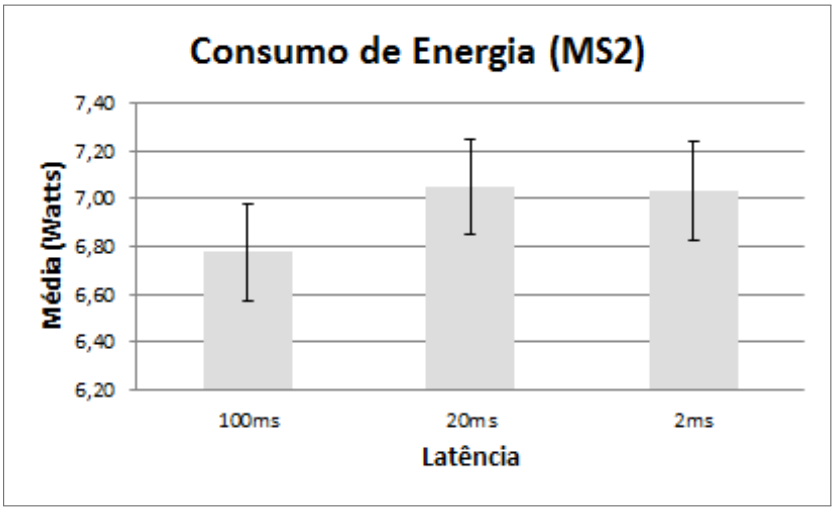

Figura 21 – Consumo Médio de Energia em MS2

menor. Com isso é possível observar que a latência impacta notavelmente no tempo do planejamento do robô, quando ela vai aumentando e quando é necessário fazer *offloading*.

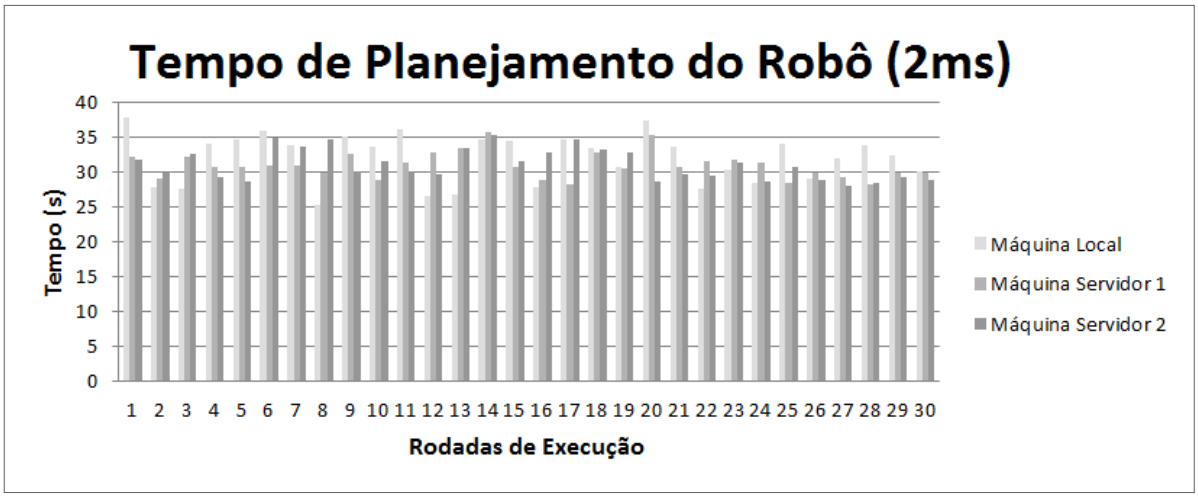

Figura 22 – Comparando o Tempo de Planejamento para 2ms

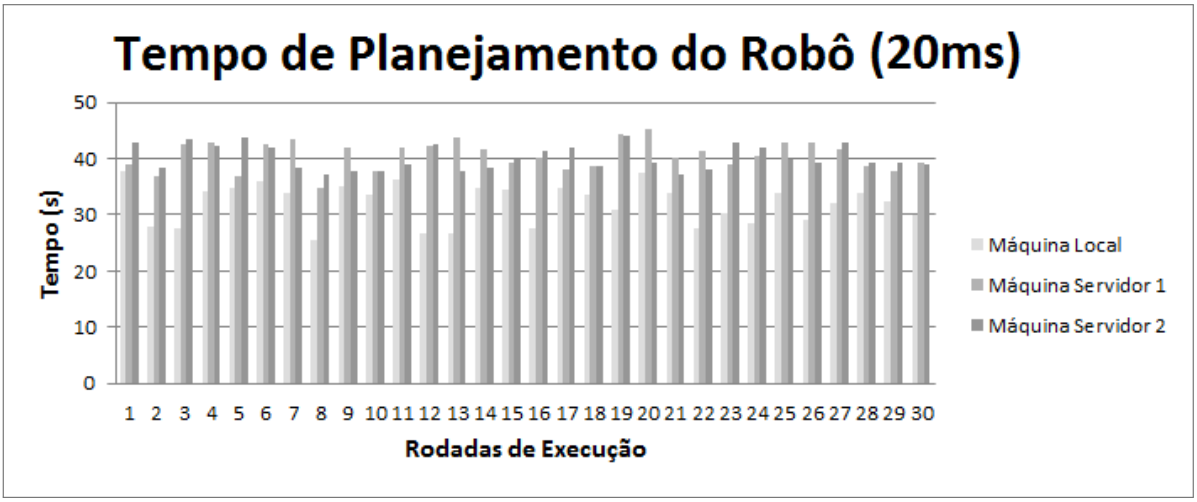

Figura 23 – Comparando o Tempo de Planejamento para 20ms

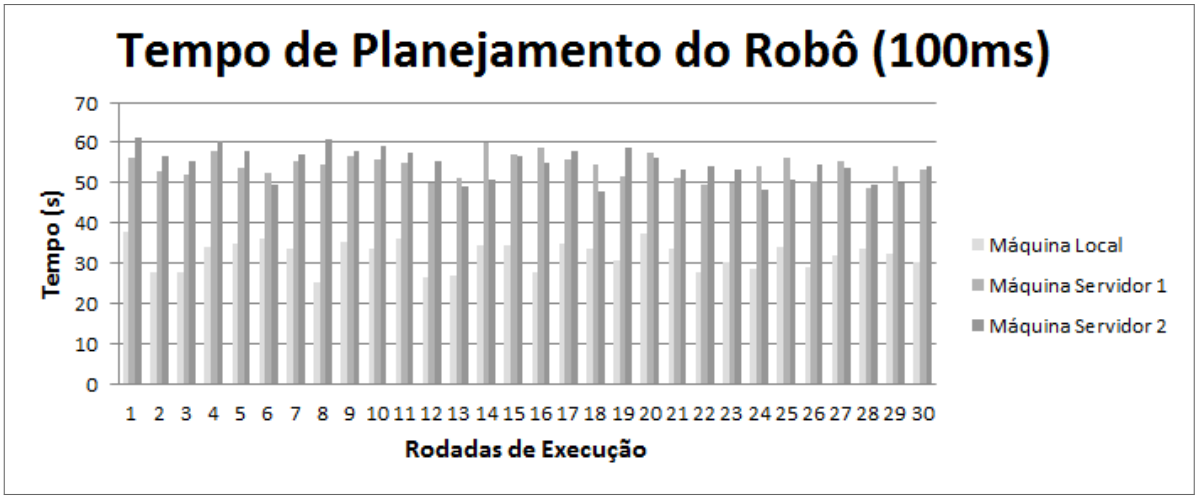

Figura 24 – Comparando o Tempo de Planejamento para 100ms

Nas Figuras 25, 26 e 27, assumindo uma distribuição normal com 95% de confiança, é apresentado o consumo médio de energia em Watts para cada máquina. Nestas figuras, observa-se que o consumo de energia ao executar a aplicação de planejamento em uma máquina local é quase o dobro de executar o *offloading*. Com isso, realizar uma tomada decisão unicamente do ponto de vista do consumo energético, seria melhor realizar o *offloading* para todos os casos. É válido notar que não se obteve nenhum pico de energia durante as execuções dos experimentos, se mantendo estável.

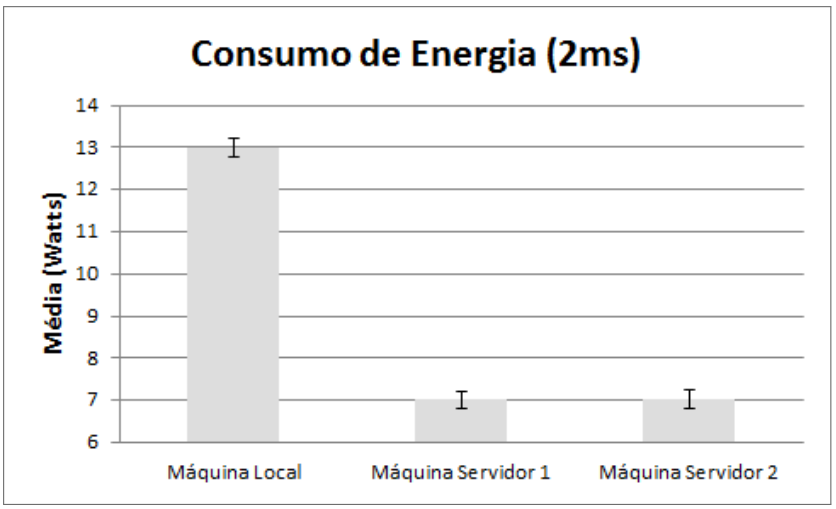

Figura 25 – Comparando Consumo de Energia para 2ms

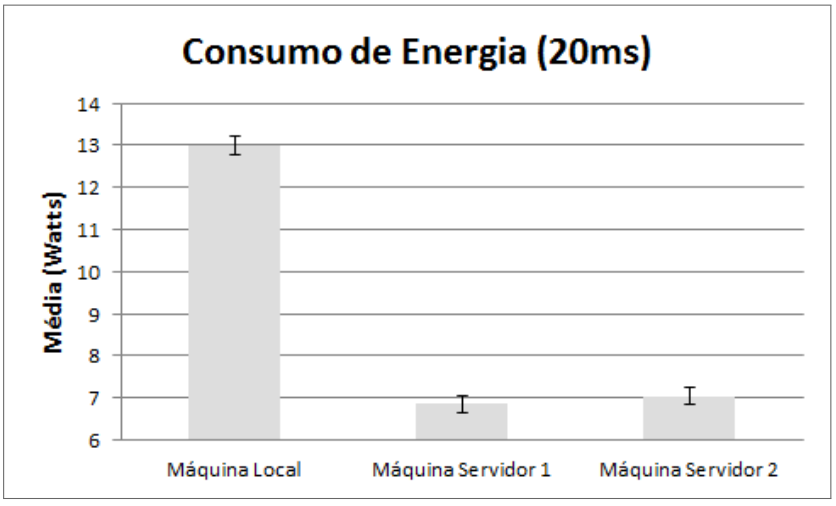

Figura 26 – Comparando Consumo de Energia para 20ms

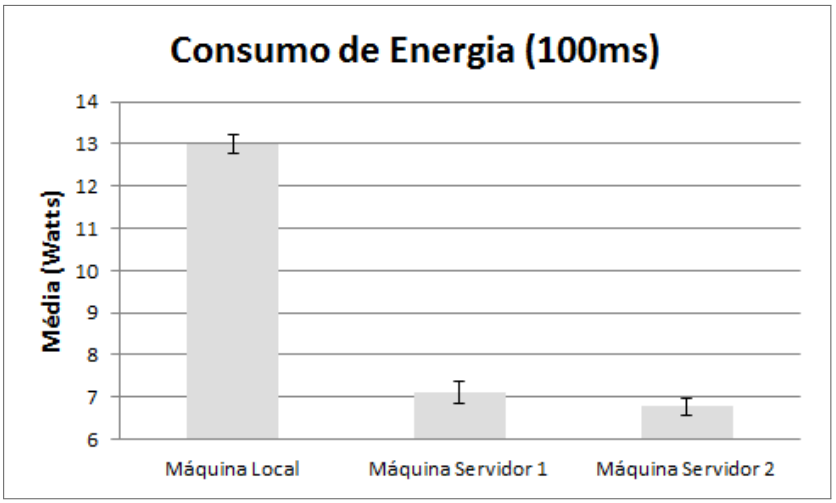

Figura 27 – Comparando Consumo de Energia para 100ms

5.3 COMPARANDO OS RESULTADOS

A partir dos resultados acima, pode-se fazer algumas comparações utilizando o mecanismo proposto neste trabalho e o mecanismo MinED descrito no Capítulo 3. Os dois mecanismos utilizam um parâmetro para definir se será feito *offloading* de processamento, com a finalidade de comparar para todos os casos, considera-se buscar o mínimo entre entre energia e tempo. É importante mencionar mais uma vez que o MinED não leva em consideração o tempo de execução de uma aplicação localmente, apenas o consumo de energia, já que eles elaboraram um mecanismo que tem como finalidade o mínimo entre energia e tempo, mas com uma prioridade para a energia. Por esse motivo, caso seja comparado os dois mecanismos, a métrica do MinED sempre sairá melhor ao executar qualquer aplicação localmente.

No entanto, foi pensado na possibilidade de algum tipo de aplicação gastar mais energia localmente do que a soma do tempo de processar no servidor e o consumo de energia local (Lembrando que, tempo e energia possuem unidades de medida diferentes e portanto não podem ser somado, contudo, foi multiplicado na Equação 4.8, energia por α com objetivo de igualar as unidades de medida, e assim correlacionar tempo e energia). Entretanto, caso a pontuação do MinED seja menor, por causa que o gasto de energia é menor, do que a pontuação do mecanismo proposto obtida pelo o tempo de execução no servidor e o gasto de energia local, a solução MinED não leva em consideração que o tempo de execução da aplicação localmente pode ser alto o suficiente para que superasse a pontuação obtida pelo mecanismo proposto, o que seria prejudicial para o desempenho da aplicação. Contudo dificilmente uma aplicação que gasta bastante energia localmente, vai possui um bom desempenho localmente também (WANG et al., 2017). Por causa disso, é levado em conta apenas o MinED em relação as máquinas servidores.

Quando feito as comparações de tomada de decisão por cada mecanismo considerando o tempo de latência de 2ms, é possível auferir a Figura 28 para a utilização do mecanismo

proposto, onde tem a pontuação final, que é a soma do consumo de energia com o tempo de processamento. Analisando está figura, é factível notar que utilizando o mecanismo proposto, em 97% dos casos, é melhor realizar *offloading* do processamento e em apenas um caso, que foi o da rodada de execução 13, realizar o processamento na máquina local seria melhor. No entanto, para o mecanismo do MinED é obtida a Figura 29. Quando comparado as Figuras 28 e 29 é possível notar que a máquina servidora que obteve um melhor desempenho, com exceção para a rodada de execução 13, onde o MinED teve a MS2 como a máquina que o obteve o melhor desempenho na Figura 29 e o mecanismo proposto obteve a ML como sendo a melhor na Figura 28. Com base nisso, pode-se afirmar que o mecanismo proposto alcançou uma melhor performance para um dos casos em comparação com o MinED e para o restante dos casos eles foram iguais.

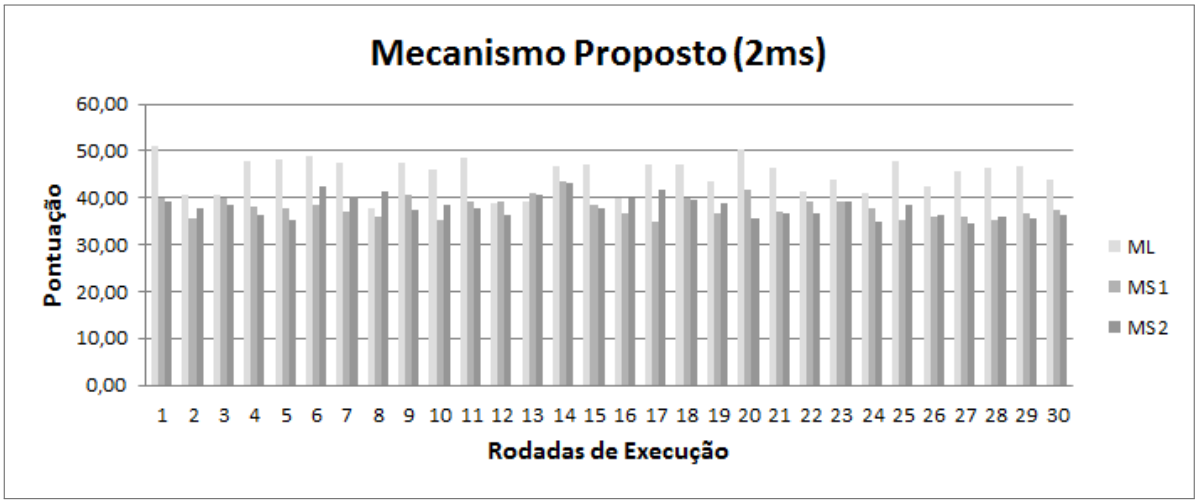

Figura 28 – Mecanismo Proposto para 2ms

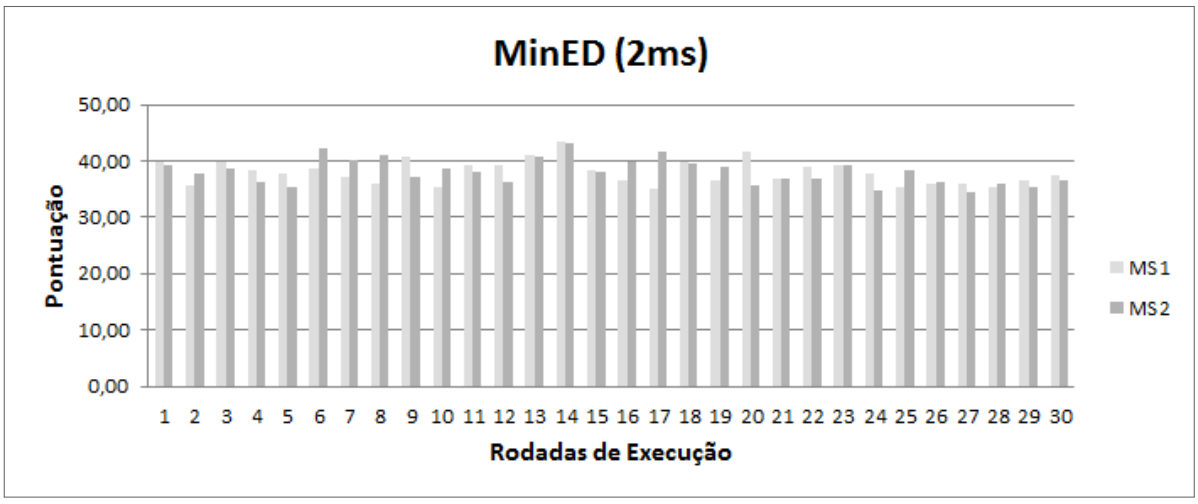

Figura 29 – Mecanismo MinED para 2ms

Em relação ao tempo de latência de 20ms, é possível obter as Figuras 30 e 31. Para o mecanismo proposto, pode-se ver que aumentou o número de casos em que executar apli-

cação localmente é melhor, no caso das seguintes rodadas de execução 2,3,4,6,8,12,13,16, 19, 22, 23, 24, 26, 27 e 30. Baseado nisso, pode-se afirmar que em 50% dos casos, executar a aplicação localmente obteria uma melhor comportamento e para todos os outros casos a máquina servidora a ser escolhida seria igual ao do MinED, como mostrado na Figura 31.

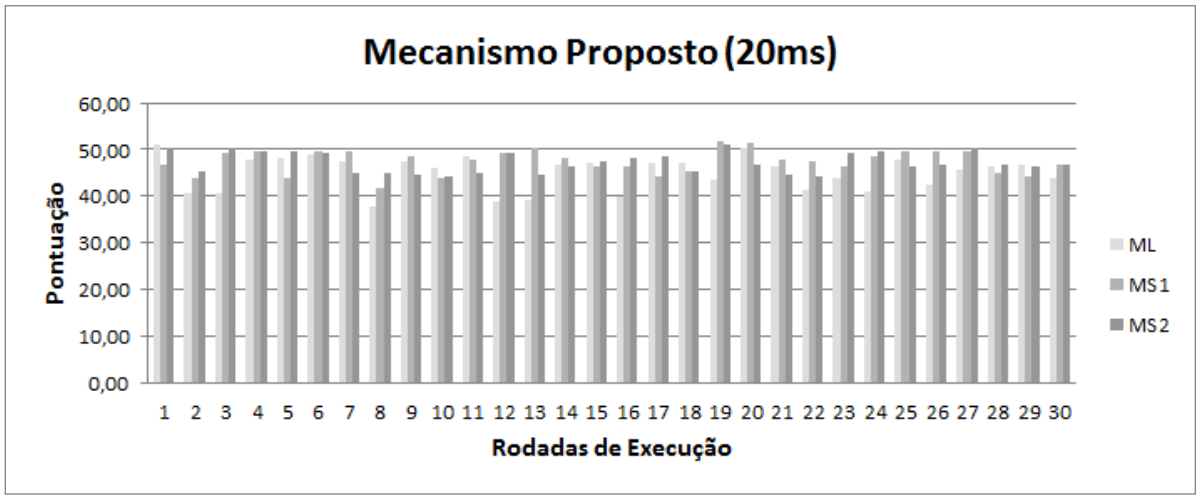

Figura 30 – Mecanismo Proposto para 20ms

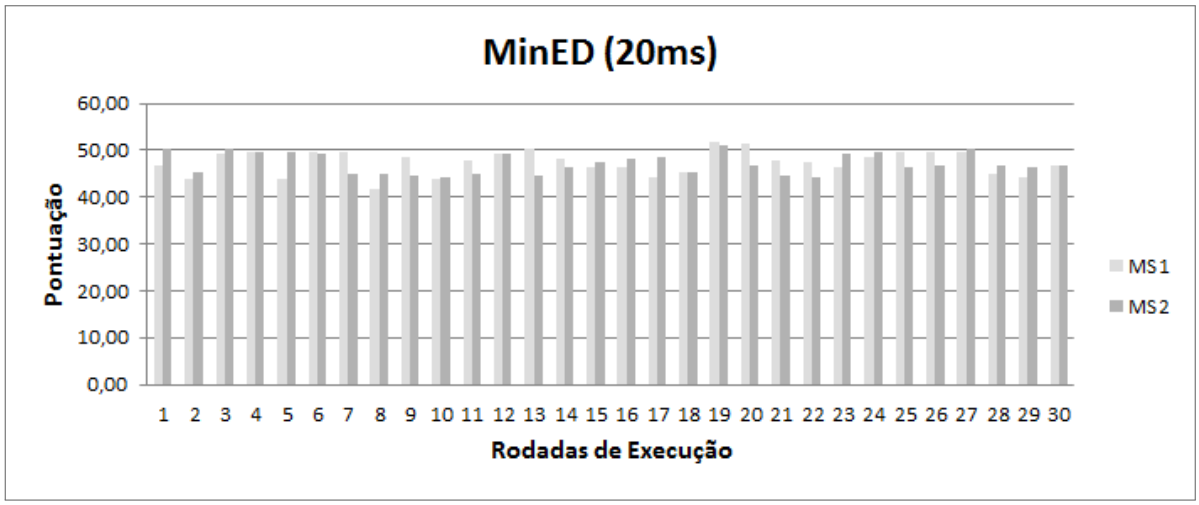

Figura 31 – Mecanismo MinED para 20ms

Já levando em conta o tempo de latência de 100ms, é possível auferir as Figuras 32 e 33. E com base nas informações dessas figuras, pode-se notar que em todos os casos executar localmente a aplicação foi melhor, consequentemente, fundamentado nos argumentos anteriores, o mecanismo proposto possui uma melhor performance sobre o MinED em todos os casos.

Dessa forma, pode-se assegurar que o mecanismo proposto neste trabalho, possui um melhor desempenho sobre o mecanismo MinED. O mecanismo proposto obteve melhor

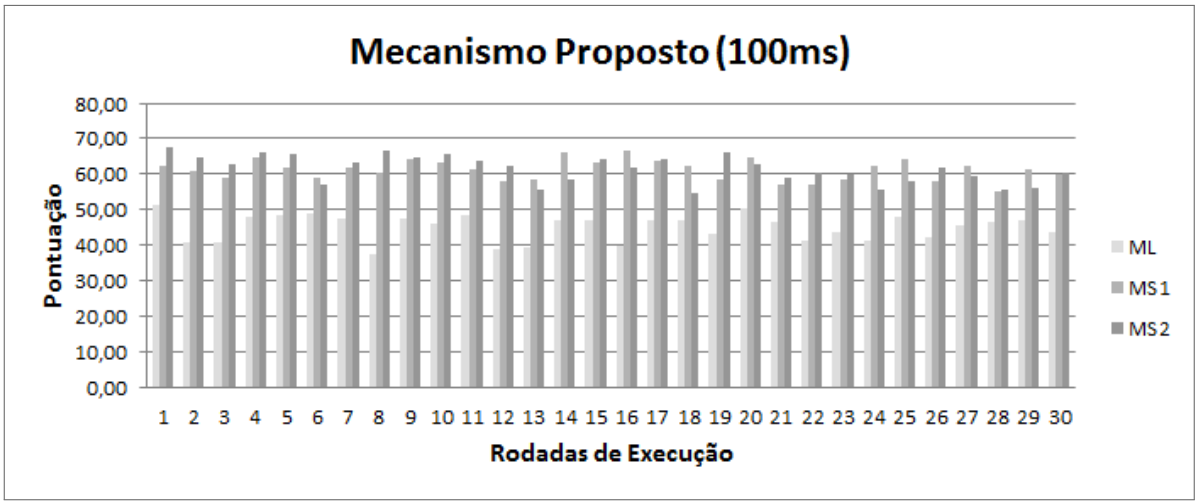

Figura 32 – Mecanismo Proposto para 100ms

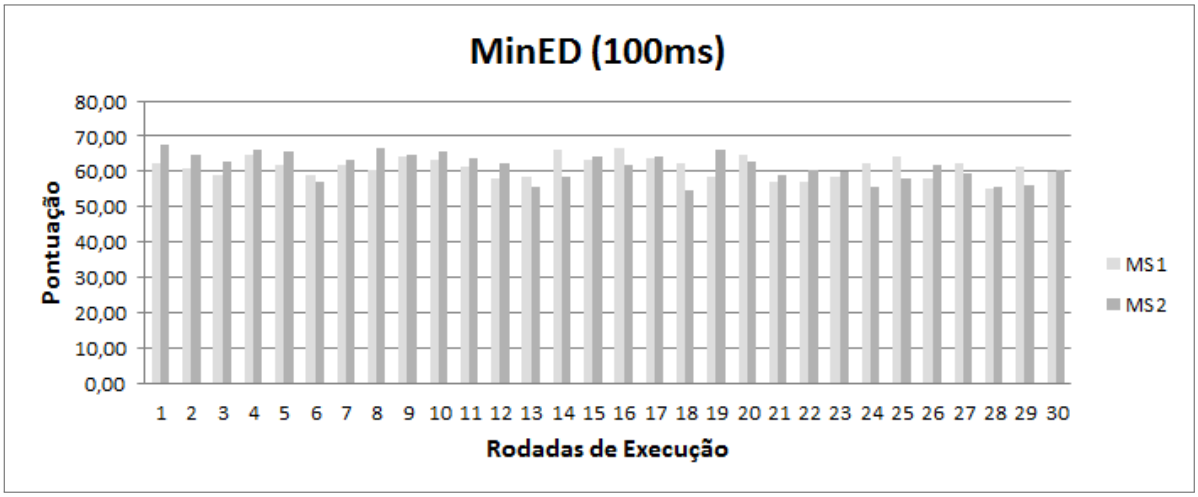

Figura 33 – Mecanismo MinED para 100ms

comportamento para os três tipos de redes, 3G, 4G e *WiFi*. Considerando a 3G, o mecanismo proposto foi 100% das vezes melhor do que o MinED, para rede 4G foi 50% e para *WiFi* foi 3%.

6 CONCLUSÃO

Com o aumento do número de aplicações robóticas mais pesadas e cada vez mais necessitando de recursos mais robustos, faz-se necessário o uso da técnica de *offloading* com a finalidade de permitir estender a capacidade de processamento dos robôs atuais. Contudo, existem mecanismos de tomada de decisão de quando se deve fazer *offloading* ou não.

Nesta dissertação, propusemos um mecanismo para tomada de decisão de *offloading* que leva em conta o tempo de processamento de uma aplicação de planejamento de movimentos do robô no contexto de cenários industriais. Com objetivo de avaliar o mecanismo proposto e comparar com outra solução da literatura, foi utilizado o cenário da competição do ARIAC em ambiente simulado e executado o algoritmo utilizado pela equipe FIGMENT na competição do ARIAC.

Através de uma metodologia baseada em mudar o tipo de rede, em que variamos as latências entre 2ms, 20ms e 100ms, utilizando uma máquina cliente executando o ambiente simulado do ARIAC e duas máquinas servidoras como recursos disponíveis para alocação de processamento, que podem medir o desempenho dos mecanismos comparado neste trabalho, tanto quanto ao tempo gasto quanto a consumo energético para executar uma aplicação localmente ou para realizar *offloading*.

A partir disso, chegamos a conclusão de que o mecanismo proposto obteve um melhor desempenho comparado com o mecanismo MinED da literatura, para as três variações de latência experimentadas neste trabalho. Para 2ms de latência, o mecanismo proposto foi 3% melhor do que o MinED e para as latências de 20ms e 100ms, foram 50% e 100% respectivamente.

Na qualidade de contribuição secundária, desenvolvemos uma aplicação de planejamento de movimentos para o braço robótico UR5 da *Universal Robots*, utilizando a arquitetura ROS. Está aplicação foi utilizada na competição de robótica ARIAC, em que a equipe FIGMENT conquistou o segundo lugar.

7 TRABALHOS FUTURO

Como trabalhos futuros, podemos investigar o quão importante o tempo e a energia são para aplicações dentro de um cenário industrial, com a finalidade de estabelecer pesos para esses fatores. Outro ponto importante seria escalonar o cenário, para investigarmos utilizando um número maior de aplicações sendo executadas dentro deste cenário industrial e aplicar o mecanismo de tomada de decisão para estas aplicações, bem como aumentar o número de servidores com diferentes recursos de capacidade de processamento com objetivo de medir a otimização da alocação de recursos. Tudo isso utilizando aplicações robóticas mais pesadas e que demandem um maior poder de processamento.

Um estudo importante que pode vir a ser feito é variar parâmetros de rede, como perda de pacotes, duplicação de pacotes ou corrupção de pacotes e avaliar o impacto disso no atraso no processamento da aplicação. Outro estudo importante seria calcular os melhores preços de robôs com certas capacidade de processamento com objetivo de redução de custos das unidades móveis utilizadas dentro da industria.

Para trabalhos futuros podemos variar a latência entre os robôs, a fim de achar a latência ideal/limite onde o mecanismo proposto sempre será melhor do que o mecanismo MinED, e também comparar o mecanismo proposto com outros mecanismo da literatura.

REFERÊNCIAS

AMAZON. *Amazon Web Services*. 2017. Disponível em: <https://aws.amazon.com/pt/s3/>. Acesso em: 28 de agosto de 2017.

ARIAC. *ARIAC*. 2017. Disponível em: <http://gazebosim.org/ariac>. Acesso em: 17 de junho de 2018.

ARUMUGAM, R.; ENTI, V. R.; BINGBING, L.; XIAOJUN, W.; BASKARAN, K.; KONG, F. F.; KUMAR, A. S.; MENG, K. D.; KIT, G. W. 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.

BCG. *The Rise of Robotics*. 2015. Disponível em: <https://www.bcg.com/publications/2014/business-unit-strategy-innovation-rise-of-robotics.aspx>. Acesso em: 02 de abril de 2018.

BONOMI, F.; MILITO, R.; NATARAJAN, P.; ZHU, J. Fog computing: A platform for internet of things and analytics. In: *Big Data and Internet of Things: A Roadmap for Smart Environments*. [S.l.]: Springer, 2014. p. 169–186.

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. Fog computing and its role in the internet of things. In: ACM. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. [S.l.], 2012. p. 13–16.

BOSCH. *Industry 4.0 – Germany takes first steps toward the next industrial revolution*. 2012. Disponível em: <https://blog.bosch-si.com/industry40/industry-40-germany-takes-first-steps-toward-next-industrial-revolution>. Acesso em: Acesso em: 02 de maio de 2018.

BRÜNGLINGHAUS, C. *Die Evolution zur Industrie 4.0 in der Produktion*. [S.l.]: Retrieved, 2015.

DEY, S.; MUKHERJEE, A. Robotic slam: A review from fog computing and mobile edge computing perspective. In: ACM. *Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services*. [S.l.], 2016. p. 153–158.

DOLUI, K.; DATTA, S. K. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In: IEEE. *Global Internet of Things Summit (GIoTS), 2017*. [S.l.], 2017. p. 1–6.

DOUGLAS, P. F. *The challenge of the Computer Utility*. [S.l.]: Addison-Wesley, Reading, Mass, 1966.

DOUKAS, C.; PLIAKAS, T.; MAGLOGIANNIS, I. Mobile healthcare information management utilizing cloud computing and android os. In: IEEE. *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. [S.l.], 2010. p. 1037–1040.

FERZLI, R.; KHALIFE, I. Mobile cloud computing educational tool for image/video processing algorithms. In: IEEE. *Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE), 2011 IEEE*. [S.l.], 2011. p. 529–533.

FORBES. *What everyone must know about industry 4.0*. 2016. Disponível em: <https://www.forbes.com/sites/bernardmarr/2016/06/20/what-everyone-must-know-about-industry-4-0/#4506ec32795f>. Acesso em: 02 de novembro de 2017.

GAZEBO. *Gazebo*. 2018. Disponível em: <http://gazebosim.org/>. Acesso em: 02 de setembro de 2018.

GOOGLE. *Network Conditions*. 2018. Disponível em: <https://developers.google.com/web/tools/chrome-devtools/network-performance/network-conditions?hl=pt-br>. Acesso em: 14 de março de 2018.

GUAN, L.; KE, X.; SONG, M.; SONG, J. A survey of research on mobile cloud computing. In: IEEE. *Computer and Information Science (ICIS), 2011 IEEE/ACIS 10th International Conference on*. [S.l.], 2011. p. 387–392.

GUO, Y.; MI, Z.; YANG, Y.; OBAIDAT, M. S. An energy sensitive computation offloading strategy in cloud robotic network based on ga. *IEEE Systems Journal*, IEEE, 2018.

HECTIC. *Powerstat: Power Consumption Calculator for Ubuntu Linux*. 2012. Disponível em: <https://www.hecticgeek.com/2012/02/powerstat-power-calculator-ubuntu-linux>. Acesso em: 15 de março de 2017.

HUANG, D. et al. Mobile cloud computing. *IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter*, v. 6, n. 10, p. 27–31, 2011.

JAZDI, N. Cyber physical systems in the context of industry 4.0. In: IEEE. *Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on*. [S.l.], 2014. p. 1–4.

KEHOE, B.; BERENSON, D.; GOLDBERG, K. Toward cloud-based grasping with uncertainty in shape: Estimating lower bounds on achieving force closure with zero-slip push grasps. In: IEEE. *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. [S.l.], 2012. p. 576–583.

KEHOE, B.; PATIL, S.; ABBEEL, P.; GOLDBERG, K. A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, IEEE, v. 12, n. 2, p. 398–409, 2015.

KUFFNER, J. Cloud-enabled humanoid robots. In: *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, Nashville TN, United States, Dec.* [S.l.: s.n.], 2010.

KUMAR, K.; LIU, J.; LU, Y.-H.; BHARGAVA, B. A survey of computation offloading for mobile systems. *Mobile Networks and Applications*, Springer, v. 18, n. 1, p. 129–140, 2013.

LUAN, T. H.; GAO, L.; LI, Z.; XIANG, Y.; WEI, G.; SUN, L. Fog computing: Focusing on mobile users at the edge. *arXiv preprint arXiv:1502.01815*, 2015.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011.

MIETTINEN, A. P.; NURMINEN, J. K. Energy efficiency of mobile clients in cloud computing. *HotCloud*, v. 10, p. 4–4, 2010.

MINAMI, Y. Executive summary: world robotics 2017 industrial robots. *Available online on https://ifr.org/downloads/press/Executive$_S$ummary$_W$R$_2$017$_I$ndustrial$_R$obots.pdf*, 2017.

MOHANARAJAH, G.; HUNZIKER, D.; D'ANDREA, R.; WAIBEL, M. Rapyuta: A cloud robotics platform. *IEEE Transactions on Automation Science and Engineering*, IEEE, v. 12, n. 2, p. 481–493, 2015.

NETEM. *Netem*. 2016. Disponível em: <https://wiki.linuxfoundation.org/networking/netem>. Acesso em: 14 de março de 2018.

RAHMAN, A.; JIN, J.; CRICENTI, A.; RAHMAN, A.; YUAN, D. A cloud robotics framework of optimal task offloading for smart city applications. In: IEEE. *Global Communications Conference (GLOBECOM), 2016 IEEE*. [S.l.], 2016. p. 1–7.

RAJPUT, H.; SAWANT, K.; SHETTY, D.; SHUKLA, P.; CHOUGULE, A. Implementation of voice based home automation system using raspberry pi. 2018.

ROBOTICS, C. *Intro to ROS*. 2015. Disponível em: <http://www.clearpathrobotics.com/assets/guides/ros/Intro\%20to\%20the\%20Robot\%20Operating\%20System.html>. Acesso em: 20 de julho de 2018.

ROS. *Robot Operating System*. 2018. Disponível em: <http://www.ros.org/about-ros/>. Acesso em: 03 de agosto de 2018.

SEGATA, M.; BLOESSL, B.; SOMMER, C.; DRESSLER, F. Towards energy efficient smart phone applications: Energy models for offloading tasks into the cloud. In: IEEE. *Communications (ICC), 2014 IEEE International Conference on*. [S.l.], 2014. p. 2394–2399.

SHIRAZ, M.; GANI, A.; KHOKHAR, R. H.; BUYYA, R. A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. *IEEE Communications Surveys & Tutorials*, IEEE, v. 15, n. 3, p. 1294–1313, 2013.

TANG, W.-T.; HU, C.-M.; HSU, C.-Y. A mobile phone based homecare management system on the cloud. In: IEEE. *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*. [S.l.], 2010. v. 6, p. 2442–2445.

WAHLSTER, W. From industry 1.0 to industry 4.0: Towards the 4th industrial revolution. In: *Forum Business meets Research*. [S.l.: s.n.], 2012.

WAIBEL, M.; BEETZ, M.; CIVERA, J.; D'ANDREA, R.; ELFRING, J.; GALVEZ-LOPEZ, D.; HÄUSSERMANN, K.; JANSSEN, R.; MONTIEL, J.; PERZYLO, A. et al. Roboearth. *IEEE Robotics & Automation Magazine*, IEEE, v. 18, n. 2, p. 69–82, 2011.

WANG, F.; XU, J.; WANG, X.; CUI, S. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. *IEEE Transactions on Wireless Communications*, IEEE, v. 17, n. 3, p. 1784–1797, 2018.

WANG, X.; WANG, J.; WANG, X.; CHEN, X. Energy and delay tradeoff for application offloading in mobile cloud computing. *IEEE Systems Journal*, IEEE, v. 11, n. 2, p. 858–867, 2017.

YANG, X.; PAN, T.; SHEN, J. On 3g mobile e-commerce platform based on cloud computing. In: IEEE. *Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference on*. [S.l.], 2010. p. 198–201.

YOU, C.; HUANG, K.; CHAE, H.; KIM, B.-H. Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, IEEE, v. 16, n. 3, p. 1397–1411, 2017.

YOUNIS, S. A.; IJAZ, U.; RANDHAWA, I. A.; IJAZ, A. Speech recognition based home automation system using raspberry pi and zigbee. *NFC IEFR Journal of Engineering and Scientific Research*, v. 5, p. 40–45, 2017.

ZHANG, G.; ZHANG, W.; CAO, Y.; LI, D.; WANG, L. Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. *IEEE Transactions on Industrial Informatics*, IEEE, 2018.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, Springer, v. 1, n. 1, p. 7–18, 2010.

ZHAO, W.; SUN, Y.; DAI, L. Improving computer basis teaching through mobile communication and cloud computing technology. In: IEEE. *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*. [S.l.], 2010. v. 1, p. V1–452.