Um Framework para Simulação de Sistemas Robóticos Baseados em Múltiplos Veículos Aéreos Não Tripulados

Gustavo Mota Simões[1,3], Alirio Santos de Sá[1,2]

[1] Laboratório de Sistemas Distribuídos (LaSiD), Instituto de Matemática e Estatística, Universidade Federal da Bahia (UFBA), Campus de Ondina, Salvador, Bahia, Brasil

gustavomota00@gmail.com, aliriosa@ufba.br

Abstract. *This paper presents a framework based on robotic middleware and a physical simulator, it provides generic functionality useful to physical simulation of scenarios with multiple unmanned air vehicles (UAVs), enabling the control of movement of said vehicles and also of the simulated time flow, in addition to obtaining information about them. The goal of that application is to act in cooperation with other software, supplying it with a layer of abstraction, facilitating the process of monitoring and manipulating the virtual environment.*

Resumo. *Este artigo apresenta um framework baseado em middleware de robótica e um simulador físico, que provê funcionalidades genéricas úteis à simulação física de cenários com múltiplos veículos aéreos não tripulados (VANT), permitindo controlar o movimento de tais veículos e o fluxo do tempo simulado, além de obter informações dos mesmos. O objetivo dessa aplicação é atuar cooperando com outros softwares, fornecendo uma camada de abstração, facilitando o processo de monitoramento e manipulação do ambiente virtual.*

1. Introdução

Os veículos aéreos não tripulados (VANTs), do inglês Unmanned Air Vehicles (UAV), também referidos na literatura como sistemas de aeronaves não tripulados, drones ou aeronaves pilotadas remotamente, têm um longo histórico de uso militar, que precede o voo pilotado por humanos, devido a sua capacidade de suporte durante o combate, como por exemplo em missões de reconhecimento [Watts et al., 2012].

Avanços na engenharia de controle e na ciência dos materiais tornaram possível o desenvolvimento de VANTs de pequena escala, dotados de câmeras e sensores. Estes VANTs de dimensões reduzidas trazem vantagens em relação aos VANTs de grande porte: podem percorrer áreas nas quais o acesso de VANTs de grande porte é ineficiente, inadequado ou impossível; possuem menor custo, o que viabiliza que vários desses VANTs sejam usados para reduzir o tempo ou aumentar as chances de sucesso (confiabilidade) de uma missão [Quaritsch et al., 2008] etc. Por conta disso, atualmente aplicações que requerem um conjunto de VANTs (multi-VANTs) capazes de colaborar ganharam um interesse crescente, como em missões de busca e salvamento, que são de tempo crítico, onde os múltiplos VANTs conseguem executá-las com menor duração se comparado a um único veículo ou métodos manuais [Dedousis e Kalogeraki, 2018]. Ademais, ter acesso a uma visualização aérea de grandes áreas, envolvendo vários

pequenos VANTs, é de utilidade em muitas aplicações onde a informação disponível é frequentemente incompleta ou inconsistente, em tais ocorrências, as câmeras e sensores ajudam na construção de uma visão geral do ambiente e na sua avaliação [Quaritsch et al., 2008]. Recentemente, VANTs são amplamente visados para aplicações como vigilância, monitoramento de desastres, gerenciamento de incêndios florestais e rastreamento [Hentati et al., 2018].

Dentre as diversas categorias, os VANTs multirotor de asas rotativas, como os quadrotores, hexarotores etc., por conta de seu alto grau de manobrabilidade, com decolagem e pouso vertical, vêm sendo usados com sucesso em aplicações comerciais, científicas e militares, envolvendo aeronaves de diversos tamanhos e em cenários distintos como, por exemplo, voos em áreas internas de edificações, entre vegetações de zonas de plantio, em dutos industriais etc. Contudo, a natureza instável do voo de VANTs multirotores pode ocasionar perda ou dano dos mesmos com facilidade, especialmente envolvendo software ou hardware protótipo [Meyer et al., 2012].

De modo geral, experimentos usando VANTs reais são custosos, de tal maneira que o desempenho desses sistemas deve ser avaliado antes de sua aplicação de fato. Por essa razão, muitos pesquisadores e projetistas recorrem à metodologia Software In The Loop (SITL), que utiliza simulação no lugar do uso de hardware. SITL se torna uma ferramenta excepcionalmente prática, pois o sistema pode apresentar defeitos durante o voo, de tal modo que se evita situações perigosas e se preserva equipamentos de custo alto de serem danificados [Hentati et al., 2018]. A experimentação em aplicações com múltiplos VANTs é particularmente desafiadora, haja vista que, além dos aspectos físicos e dinâmicos envolvendo o controle de movimentação de cada VANT individual, é necessário considerar os aspectos de comunicação e cooperação entre os VANTs envolvidos de modo a cumprir adequadamente os objetivos da missão. Além disso, a programação de multi-VANT é uma área nova, caracterizada pela carência de frameworks disponíveis [Dedousis e Kalogeraki, 2018].

Neste contexto, o objetivo deste trabalho é introduzir um framework que facilite a realização de experimentos, por meio de simulações com aplicações distribuídas baseadas em multi-VANT, viabilizando que os aspectos físicos, de controle de movimentação, de comunicação em rede e de coordenação entre VANTs sejam verificados em uma mesma simulação. O framework proposto estende as facilidades de um middleware voltado à robótica, o Robot Operating System (ROS) e de um simulador de física, o Gazebo, além de oferecer interfaces para integração com outros softwares, de interesse particular são os simuladores de redes de computadores. Em seu estado atual, o foco está na simulação de VANTs, porém ROS é um middleware adequado para Hardware In The Loop (HITL), de forma que pode ser incrementado para tal.

As seções do presente texto estão organizadas como segue. A Seção 2 apresenta brevemente trabalhos relacionados, a Seção 3 discute o framework e sua estrutura, a Seção 4 traz resultados do custo de simulação utilizando o framework proposto, e a Seção 5 faz as considerações finais, incluindo trabalhos futuros e em andamento.

2. Trabalhos Relacionados

[Hentati et al., 2018] faz uma revisão geral das características de simuladores e outras ferramentas utilizadas na avaliação de desempenho de sistemas com VANTs. Esse

trabalho apresenta comparações entre as ferramentas abordadas, evidenciando a adequação das mesmas para cada funcionalidade. [Dedousis e Kalogeraki, 2018] apresenta um framework para programar enxames ou unidades isoladas de VANTs intitulado PaROS, voltado para missões genéricas, tanto simuladas quanto reais. Esse trabalho utiliza um conceito de enxames abstratos, que é discutido no mesmo. Tal framework possui interface de programação em Java. São apresentados resultados em termos de tempo de execução de missões simuladas e reais. [Grabe et al., 2013] também traz um framework modular e extensível baseado em ROS para controle de múltiplos VANTs quadrotor, intitulado TeleKyb. O framework possibilita interface com humanos, integração com o software MATLAB/Simulink da MathWorks, utilização de VANTs reais ou simulados e lidar com obstáculos. O trabalho apresenta resultados de experimentos com trajetória, atitude, enxames e Human In The Loop. Contudo, esses trabalhos não consideram a necessidade de avaliar diferentes aspectos de comunicação e cooperação multi-VANT nos cenários de aplicação simulado.

3. Descrição e Estrutura do Framework Proposto

3.1. Descrição de Aspectos Arquiteturais e Decisões de Projeto

O framework proposto é disponibilizado na forma de um pacote de software, escrito em C++, intitulado drone_middleware, desenvolvido sobre a plataforma ROS (Robotic Operating System) [Open Robotics, 2020], a qual oferece um middleware de comunicação que permite a integração de diferentes sistemas robóticos e viabiliza que o framework proposto também possa ser usado na prototipagem e simulação usando dispositivos físicos reais. Dentre os padrões arquiteturais de comunicação disponibilizados pelo ROS, o pacote drone_middleware faz uso do padrão Editor/Assinante (Publisher/Subscriber) [Steen e Tanenbaum, 2017], o qual permite que dispositivos físicos reais ou simulados troquem informações, de forma desacoplada, envolvendo tópicos pré-definidos - i.e., toda vez que um nó (Editor), por exemplo sensor, VANT etc., divulga informações sobre um tópico específico, todos os demais nós (Assinantes) que se inscreveram neste tópicos, por exemplo atuadores, controladores, VANT etc., recebem a informação.

Dentre os simuladores de física existentes, o Gazebo [OSRF, 2020] foi escolhido, pois é amplamente utilizado para a simulação de sistemas robóticos e é facilmente integrado à infraestrutura do ROS - o que permite que dispositivos físicos (sensores, atuadores, controladores etc.) e módulos de software externos interajam, fornecendo e consumindo informações, com elementos do ambiente simulado.

O framework drone_middleware abstrai as plataformas ROS e Gazebo e fornece interfaces ao programador de sistemas de VANTs, que lhe permitem simular cenários diversos, de maneira programática e automatizada, e colher dados em tempo de execução para utilização integrada em outros softwares (simuladores de rede, por exemplo). Por outro lado, o framework fornece os mecanismos necessários para que simuladores a eventos discretos para redes de computadores (e.g., Omnet++ [Varga e Hornig, 2008]) tenham as suas ações sincronizadas com o simulador de física, o que implica que os relógios de simulação dos simuladores envolvidos precisam ser sincronizados. Isto é realizado via Chamada Remota de Procedimentos (RPC) [Steen e Tanenbaum, 2017], de modo a (ver Figura 1): ter um custo menor que o uso da

infraestrutura do ROS; usar uma API que padronize o acesso, exponha as funcionalidades principais e oculte a complexidade interna dos simuladores de rede.

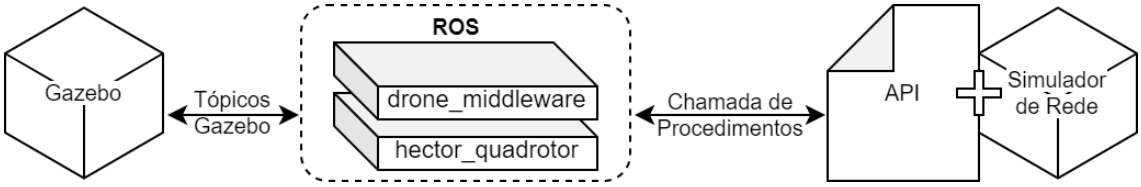


Figura 1. Arquitetura simplificada da solução proposta, as setas representam canais de comunicação, os dois pacotes ROS presentes se comunicam.

3.2. Discussão dos Principais Módulos

O drone_middleware é organizado em módulos, instanciados em nós da infraestrutura do ROS. Dentre os principais módulos implementados, se destacam os módulos PoseListener, CoreTeleoperator, Teleoperator e WorldController (ver Figura 2).

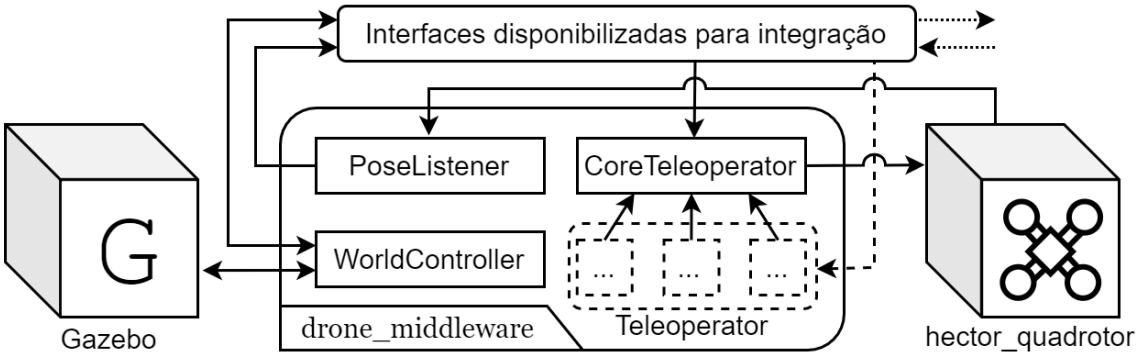


Figura 2. Diagrama simplificado do pacote drone_middleware, setas e suas direções representam canais e suas direções de comunicação, contornos tracejados indicam classe abstrata, setas pontilhadas são interfaces disponíveis para outros softwares.

O módulo da classe PoseListener se conecta nas interfaces fornecidas pelas instâncias de VANTs implementadas no simulador de física usado, por exemplo, o pacote hector_quadrotor. Durante as simulações, as posições dos VANTs são monitoradas e armazenadas em estruturas de dados. A qualquer momento essa informação pode ser fornecida através de requisições.

O módulo da classe CoreTeleoperator também se conecta nas interfaces dos VANTs, mas para emissão de comandos, esse módulo armazena de forma estruturada e ordenada todos os comandos em andamento e os que devem ser enviados em instantes futuros para os VANTs. É possível escalonar comandos em sua estrutura através de requisições. Os módulos do tipo Teleoperator são instâncias de classe abstrata, que representam quaisquer objetos capazes de determinar comandos que serão escalonados para os VANTs, independentemente do modo como tais instruções são obtidas ou geradas. Essa classe padroniza a interação com CoreTeleoperator, permitindo a construção de novos módulos, que gerem ou recebam comandos por meios não implementados, como por exemplo um controlador remoto ou um servidor de rede.

O módulo da classe WorldController, além de interagir nos canais do ROS, se comunica também diretamente com o Gazebo, permitindo controlar aspectos da física do mundo simulado e da progressão do tempo. É possível indicar ao simulador que ignore o tempo real e transcorra tão rápido quanto possível, ou que execute passos discretos no tempo, de modo a viabilizar a sincronização do tempo simulado com o utilizado por outros softwares que estejam sendo executados em conjunto na simulação.

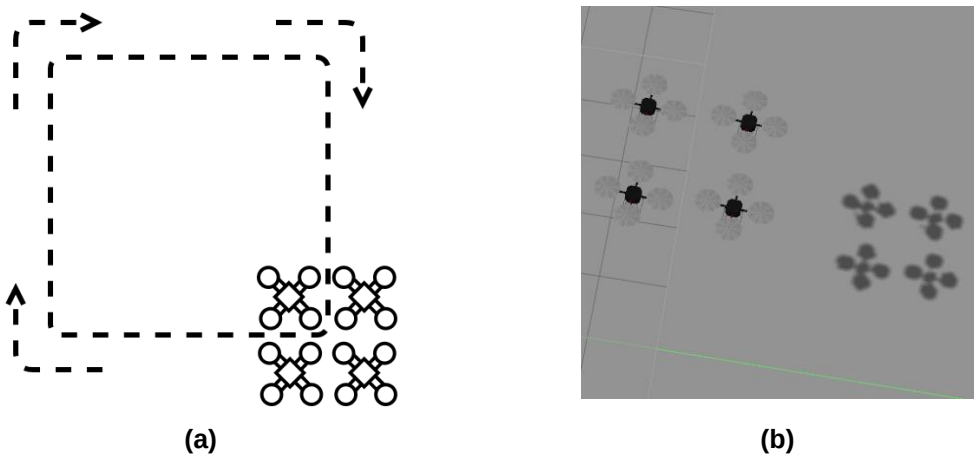
Toda operação dependente do tempo é baseada no tópico ROS "/clock", que é publicado pelo Gazebo, que por sua vez é controlado por WorldController, portanto este controla a cadência do conjunto. Esse último armazena informações sobre a simulação, como tempo total, tempo decorrido do passo atual e instante inicial, em unidades reais e simuladas, que podem ser disponibilizadas por meio de requisições a qualquer instante.

O framework também disponibiliza um ferramental (scripts em Python), que automatiza o processo de iniciar uma simulação física no Gazebo, permitindo introduzir nela um número arbitrário de VANTs em posições determinadas e iniciar os módulos principais do drone_middleware, usando parâmetros definidos em linha de comando.

Apesar de o loop de controle parecer interrompido na Figura 2, isso se deve ao fato da estratégia de controle específica ficar a cargo do programador, que pode estender a classe Teleoperator ou trabalhar fora do framework, utilizando apenas suas interfaces.

4. Resultados

A metodologia escolhida para estimar o desempenho do framework é a avaliação do seu custo em recursos consumidos da máquina que executa a simulação (percentual consumido de CPU e RAM). Foram elaborados cenários de teste, com 1 a 20 VANTs, em que estes se deslocam, em formação, a velocidade constante, dando voltas repetidas vezes ao longo dos lados de um quadrilátero imaginário. Cada cenário é executado até atingir 60 segundos de tempo simulado, com todos os módulos do drone_middleware em execução. Os parâmetros avaliados foram o tempo real gasto para executar 60 segundos simulados e o uso percentual médio de processamento da CPU e espaço em memória utilizados durante cada simulação, relativo ao total disponível na máquina:
CPU Ryzen 5 2400G, 4 núcleos, 8 threads, 3.6GHz, RAM 16GB, DDR4, 2400MHz.



(a) (b)

Figura 3. (a) Representação em diagrama do cenário proposto, as setas representam a trajetória cíclica a ser seguida. (b) Captura de tela do simulador que mostra os VANTs durante a execução do cenário.

O controlador utilizado foi uma implementação de Teleoperator que comanda os VANTs conforme uma movimentação pré-determinada, na forma de uma lista de comandos armazenados em um arquivo de texto em formato apropriado, de modo que a mobilidade possuía sempre o mesmo comportamento para um dado número de VANTs.

Para geração de dados de tempo, cada um dos cenários é executado 10 vezes, em seguida os tempos real e simulado decorridos são gravados pelo módulo da classe WorldController, para o consumo percentual de processador e memória da máquina, os cenários são executados 5 vezes, e valores são coletados via script, a intervalos de meio segundo, com uso do comando Unix "top". Esses dados foram armazenados em arquivos de texto e processados para gerar valores médios, tanto para tempo de execução quanto consumo de recursos em cada situação. Todos os testes foram realizados na mesma máquina. Não foram utilizadas interfaces gráficas ou quaisquer outras ferramentas durante a simulação. O sistema operacional utilizado foi o Xubuntu versão 16. Durante as simulações, foi considerado apenas o período entre o disparo do relógio da simulação e sua interrupção na marca dos 60 segundos simulados, o tempo necessário para executar scripts que fazem configurações, preparativos ou inicialização de ferramentas de software e inclusão de VANTs na simulação não foi contabilizado.

A Figura 4 mostra que, abaixo de 8 VANTs, o simulador equipado com o framework proposto simula próximo do tempo real. Os dados indicam um crescimento não linear do tempo de execução com o aumento do número de VANTs simulados, porém no pior caso inferior a 10 segundos por VANT. Quanto ao consumo de recursos, dados são apresentados em valores percentuais na Figura 5 e normalizados na Figura 6, tomando como referência o caso de único VANT, todos os valores da Figura 6 são fatores multiplicativos em relação a este - i.e., no caso com 20 VANTs o uso de memória foi cerca de duas vezes e meia o do caso base.

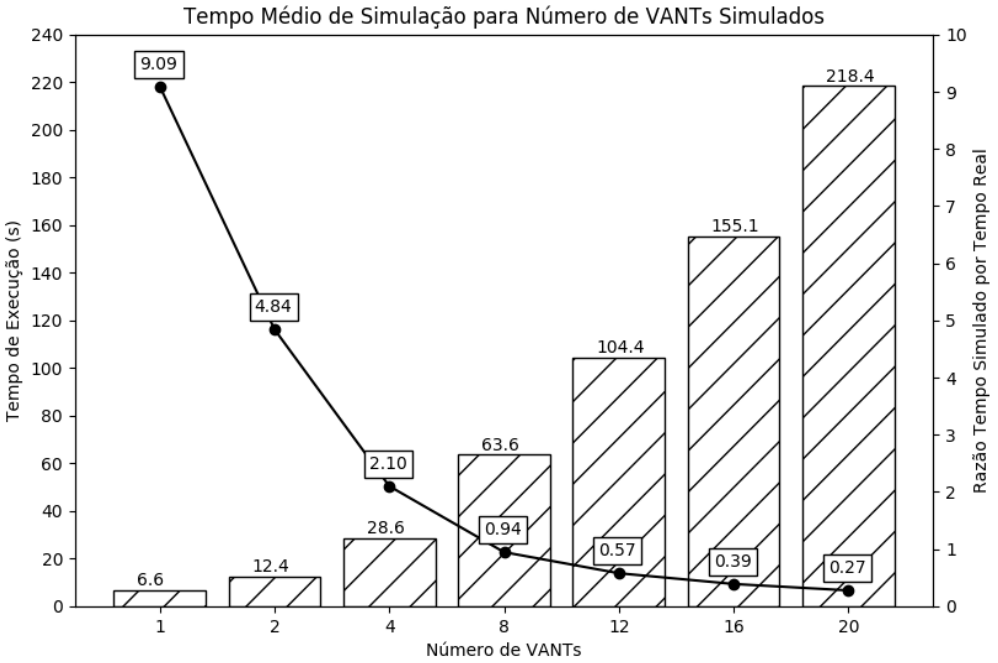


Figura 4. Tempo médio necessário para executar 60 segundos de tempo simulado, para cada cenário, em barras o tempo de execução, em linhas a razão entre os segundos simulados e os segundos reais.

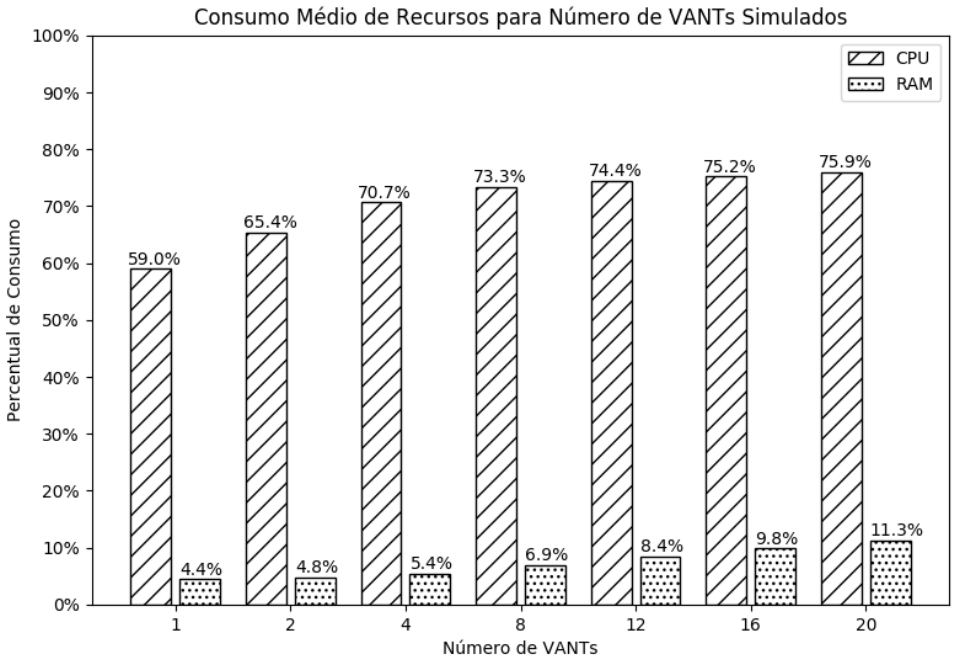


Figura 5. Consumo médio percentual de recursos durante 60 segundos virtuais, em barras tracejadas o consumo de CPU, em barras pontilhadas o consumo de RAM.

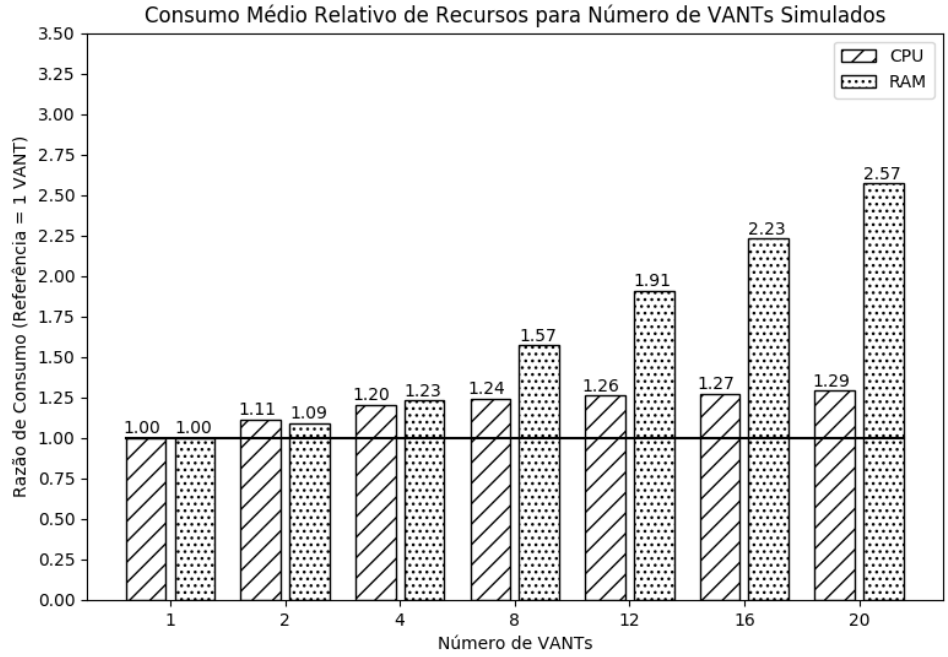


Figura 6. Consumo médio relativo de recursos durante 60 segundos virtuais, em barras tracejadas o consumo de CPU, em barras pontilhadas o consumo de RAM, em linha sólida a referência - i.e., consumo com 1 VANT (ver Figura 5).

5. Considerações Finais

Este artigo traz resultados parciais de um trabalho de pesquisa de iniciação científica na graduação, parte de um projeto em andamento, que objetiva integrar um ambiente de

simulação de protocolos de comunicação, para redes veiculares, a uma plataforma de experimentação para missões multi-VANTs. Foi apresentado um framework que estende as plataformas ROS e Gazebo, para fornecer interfaces que permitam simular aplicações multi-VANTs em cenários diversos, de maneira programática e automatizada e colher dados em tempo de execução para utilização integrada em outros softwares.

Tal framework oferece facilidades de integração com simuladores de redes de computadores, de maneira que protocolos de comunicação, propriedades físicas, controle de movimentação e cooperação entre VANTs possam ser avaliados simultaneamente. Apesar das interfaces implementadas já contribuírem para viabilizar a integração entre simuladores, algum esforço ainda é necessário no desenvolvimento de API para simulador específico. Assim, avaliar e validar melhor este aspecto de integração ainda é uma proposta de trabalho futuro. Outra proposta interessante de trabalho futuro seria a utilização de Hardware In The Loop nos experimentos.

Referências

DEDOUSIS, D.; KALOGERAKI, V. A Framework for Programming a Swarm of UAVs. Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference on - PETRA '18, 2018.

GRABE, V. et al. The TeleKyb framework for a modular and extendible ROS-based quadrotor control. 2013 European Conference on Mobile Robots, 2013.

HENTATI, A. et al. Simulation Tools, Environments and Frameworks for UAV Systems Performance Analysis. 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), 2018.

MEYER, J. et al. Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo. Simulation, Modeling, and Programming for Autonomous Robots, p. 400-411, 2012.

OPEN ROBOTICS. ROS: Robotic Operating System, 2020, Disponível em: https://www.ros.org/. Acesso em: 27 mar. 2020.

OSRF. Open Source Robotics Foundation. Gazebo: Robot Simulation Made Easy, 2020a. Disponível em: http://gazebosim.org/. Acesso em: 27 mar. 2020.

QUARITSCH, M. et al. Collaborative microdrones: applications and research challenges. Proceedings of the 2nd International ICST Conference on Autonomic Computing and Communication Systems, 2008.

STEEN, M. V.; TANENBAUM, A. S. Distributed Systems: Principles and Paradigms, 3rd ed., 2017, 704p. Disponível em: https://www.distributed-systems.net/index.php/books/ds3/. Acesso em: 27 mar. 2020.

VARGA, A.; HORNIG, R. An Overview of the OMNeT++ Simulation Environment. Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, 2008.

WATTS, A.; AMBROSIA, V.; HINKLEY, E. Unmanned Aircraft Systems in Remote Sensing and Scientific Research: Classification and Considerations of Use. Remote Sensing, v. 4, n. 6, p. 1671-1692, 2012.