Capítulo 3

Emulando Redes de Comunicação para Sistemas de Múltiplos Drones: uma abordagem inicial

Diego S. Pereira, Luís B. P. Nascimento,
Vitor G. Santos, Pablo J. Alsina

Abstract

The popularization of applications that use Unmanned Aerial Vehicles (UAVs) has greatly stimulated research development in this area. Applications with multiple UAVs demand a network capable of enabling communication between the aircraft fleet and the base station on the ground. However, it is not simple to build this type of configuration in real life, making virtual environments an important tool to simulate this scenario with accuracy. In this context, the emulated environment of the Mininet-WiFI integrated with the Robotic Simulation Platform V-REP offers an interesting alternative to represent this scenario. Thus, this mini course is proposed to present how to set up some scenarios for carrying out experiments with multiple drone systems applications.

Resumo

A popularização de aplicações que fazem uso de Veículos Aéreos Não Tripulados (VANTs) tem estimulado o desenvolvimento de pesquisas nessa temática. Aplicações com múltiplos VANTs demandam uma rede capaz de viabilizar a comunicação entre todos os elementos da frota de aeronaves e a estação base em terra. Dessa forma, é preciso fomentar ambientes e estratégias capazes de representar esse cenário de maneira mais próxima da realidade. Nesse contexto, a utilização do ambiente emulado do Mininet-WiFI integrado à Plataforma de Simulação Robótica V-REP apresentam-se com uma alternativa interessante para essa representação. Dessa forma, é proposto um minicurso que visa apresentar essa integração e como montar os primeiros cenários para realização de experimentos com aplicações de sistemas de múltiplos drones.

3.1. Introdução

Os Veículos Aéreos Não Tripulados (VANTs), popularmente conhecidos como drones, vêm sendo utilizados em diversas atividades, tais como inspeção, mapeamento, transporte

de cargas, entre outras [Hong et al. 2021]. Apesar de muitos avanços, existem alguns fatores que limitam a utilização dessas aeronaves em algumas situações, destaca-se a restrição de *payload* (capacidade de transporte de carga útil), tempo de voo (limitada a bateria embarcada) e o alto custo (motores, sensores, atuadores, etc.).

Dessa forma, pesquisas têm investido no uso de múltiplos drones atuando de forma cooperativa para superar tais limitações. Chamados de Sistemas Multi-VANT, eles agregam escalabilidade, flexibilidade, resiliência e autonomia para realização de missões complexas a partir da divisão de tarefas [Fu et al. 2019]. São exemplos interessantes de aplicações que tomam proveito dos benefícios providos por um sistema multi-VANT: o trabalho de [Silva et al. 2019] para realizar varredura grandes regiões em alto mar; o trabalho de [Santos et al. 2019] que tem como objetivo monitorar regiões de proteção ambiental; os autores em [Bai et al. 2021] ofertam serviços de comunicação para usuário em terra; e o trabalho de [Wang et al. 2021] faz uso de um sistema multi-VANT para auxiliar na navegação de grandes embarcações.

Para o funcionamento adequado de um sistema multi-VANT é fundamental a constituição de uma rede de comunicação apta a assegurar a troca de informações entre as aeronaves que integram o sistema. Essa rede de comunicação é denominada Rede *Ad Hoc* Aérea (*Flying Ad Hoc Network* - FANET). Uma FANET é criada de maneira *ad hoc* e deve garantir que todos os nós da rede possam comunicar-se entre si e com a estação base (EB) em terra. Para tal, uma aeronave deve permanecer, obrigatoriamente, no alcance de comunicação de outra e, pelo menos, uma delas necessita ter conectividade com a EB. As FANETs diferem de outros tipos de rede *ad hoc* na manutenção da conectividade, movimentação dos nós, na forma de entrega de dados, descoberta de serviços, entre outras características [Chriki et al. 2019].

O comportamento dinâmico inerente às FANETs, em consequência dos movimentos das aeronaves e da própria natureza da comunicação sem fio, inserem desafios em seu gerenciamento, afinal é imprescindível assegurar um conjunto de requisitos para não inviabilizar a comunicação entre as aeronaves. Nesse cenário, os avanços nos sistemas de telecomunicações, nos protocolos e interfaces de comunicação vêm viabilizando o desenvolvimento das FANETs. Tecnologias importantes providas pelo 4G e 5G, além de avanços com a família IEEE 802.11, são itens importantes para permitir o funcionamento de FANETs. Outras tecnologias como IEEE 802.15.4, DigiMesh e ZigBee também são utilizadas como alternativas viáveis conforme a aplicação na qual a FANET está inserida [Pereira et al. 2020].

Dentro desse contexto, pesquisas relacionadas à temática de múltiplos drones enfrentam alguns desafios para realizar experimentos e avaliar suas proposições, entre eles o alto custo para aquisição de aeronaves, demanda por equipamentos para prover infraestrutura para execução do experimento, além da força de trabalho humano na montagem, durante e finalização dos testes, são exemplos de dificuldades encontradas para experimentação nesse área. Dessa forma, é preciso buscar alternativas que tornem viáveis a execução de ensaios simulados e emulados. Uma opção interessante é a integração do Mininet-WiFi [Fontes et al. 2015] com o CoppeliaSim[1], uma plataforma para simulação de robôs [Rooban et al. 2021].

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

A Figura 3.1 apresenta essa integração em funcionamento. É possível ver o terminal (a esquerda) e a representação gráfica dos nós (a direita) do emulador Mininet-WiFi. Enquanto as três aeronaves, do tipo quadrirrotor, no CoppeliaSim (ao centro). Dessa forma, é possível criar cenários de forma rápida e com uso de *drivers* reais, dada a natureza de emulação proposta pelo Mininet-WiFi. Além disso, é possível variar a quantidades de aeronaves e empregar diferentes tipos de tecnologias, permitindo mais flexibilidade ao pesquisador na construção da emulação.

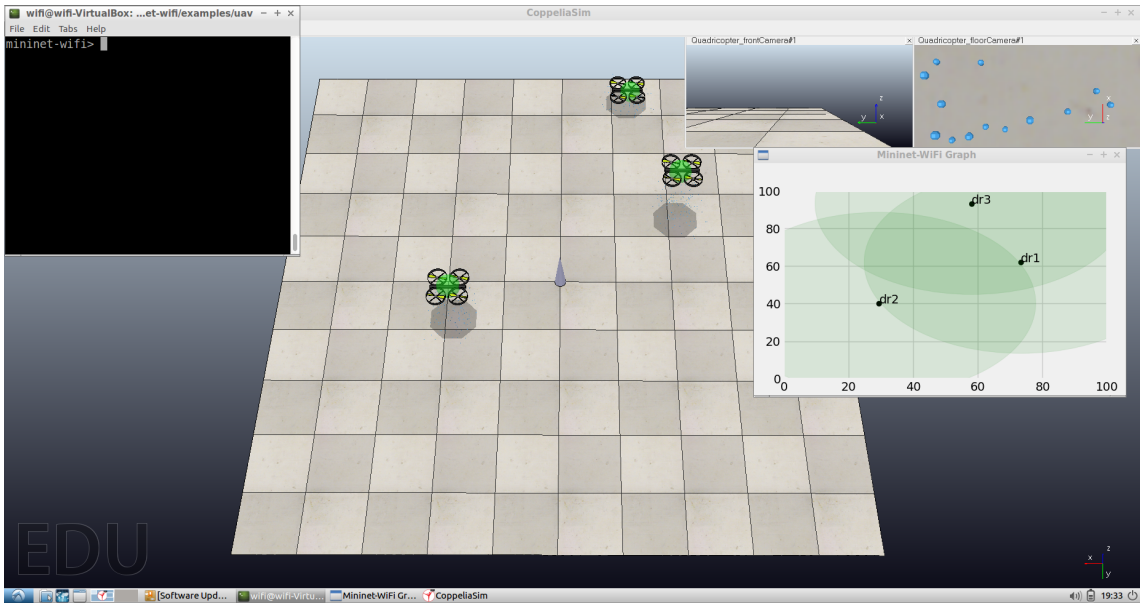

Figura 3.1. Ambiente virtual em execução (Mininet-WiFi e CoppeliaSim).

Portanto, é possível perceber que existem alternativas para investigar as redes de comunicação utilizadas em sistemas de múltiplos drones. Além disso, após a construção do conhecimento básico para criação e utilização dos cenários, agrega-se muito mais valor e motivação a pesquisa, dada a característica visual do CoppeliaSim.

Nessa perspectiva, este capítulo tem como objetivo principal apresentar uma estratégia para emular redes de comunicação para sistemas de múltiplos drones em um ambiente totalmente virtual. Para tal, será feita uma introdução aos principais conceitos referentes a sistemas multi-VANT, além da apresentação de algumas tecnologias de comunicação sem fio utilizadas para viabilizar a comunicação entre as aeronaves desse tipo de sistema. Também serão feitas demonstrações de avaliação de performance com uso de ferramentas abertas(*open source*) a partir de métricas frequentemente utilizadas.

O conteúdo trata-se de uma visão introdutória. Contudo, já permite criar ambientes virtuais customizados para realização de diversas investigações. Ademais, estimular e difundir pesquisas nessa temática, buscando fomentar novos colaboradores. A seguir, será feita uma rápida apresentação sobre redes de comunicação de sistemas de múltiplos drones. Após isso é feito um detalhamento do emulador Mininet-Wifi e da sua integração com o CoppeliaSim. Em seguida, métricas para avaliação de performance de rede são listadas, bem como algumas ferramentas. Por fim, é feito um estudo de caso agregando todo o conteúdo.

3.2. Redes de Comunicação para Sistemas de Múltiplos Drones

Como descrito na introdução do capítulo, um sistema multi-VANT é composto por aeronaves e por, no mínimo, uma estação base. A estação base, normalmente, está em solo e atua como a interface entre o operador e o sistema. Os comados têm como origem a EB e, quando necessário, as aeronaves enviam dados da aplicação para validação do operador. Por exemplo, ao detectar um barco em uma região de navegação proibida, o VANT deve enviar imagens da embarcação para averiguação do operador. Portanto, é necessário garantir a continuidade do enlace de comunicação entre a FANET (composta pelos VANTs) e pela EB.

A Figura 3.2 ilustra o exemplo citado anteriormente. Uma frota de VANTs realizando inspeção de uma região em alto mar. Para tal, as aeronaves são equipadas com uma tecnologia de comunicação sem fio para construção da FANET. Durante toda a operação do sistema multi-VANT ocorre troca de informações entre os VANTs para garantir a execução da missão. Essas informações são oriundas da própria aplicação responsável em controlar o sistema e viabilizar a cooperação entre as aeronaves. Exemplos de funções dessa aplicação são podem ser: manter a comunicação contínua entre as aeronaves e a EB, controlar a posição dos VANTs, ações de recuperação em caso de falhas, entre outras.

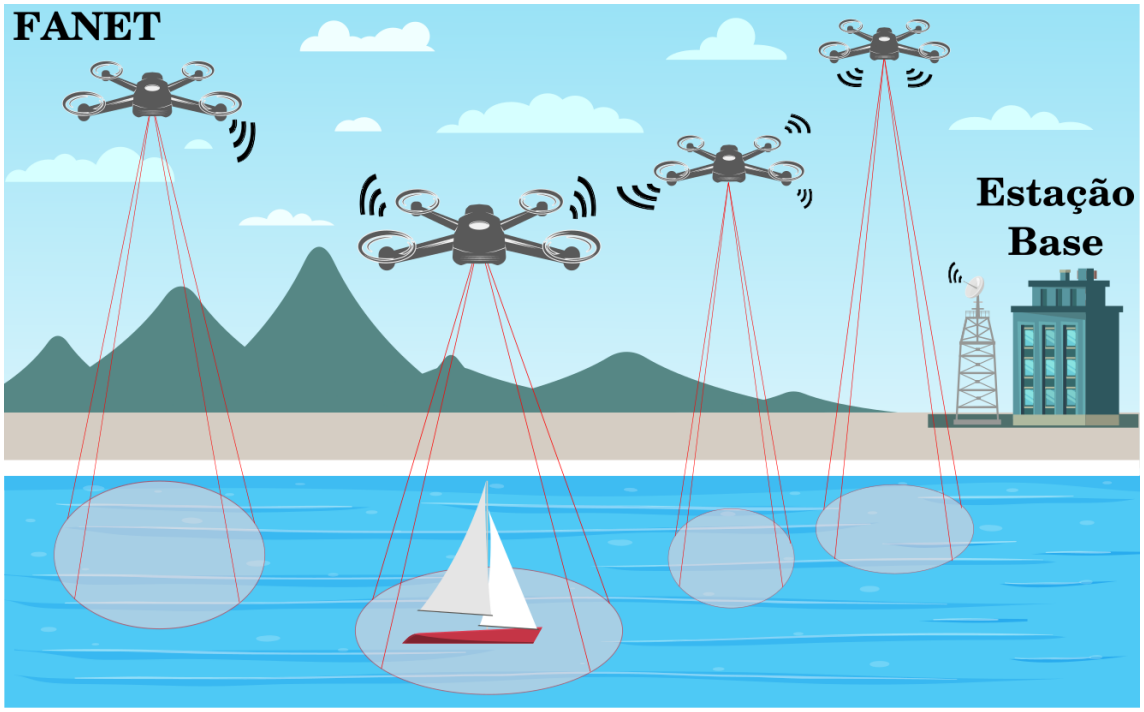

Figura 3.2. Exemplo de sistema multi-VANT executando inspeção em alto mar.

3.3. Emulação de Redes Sem Fio com Mininet-WiFi

A emulação de redes sem fio é uma tarefa árdua e exige um conjunto de softwares integrados que permitam a construção de cenários capazes de reproduzir o comportamento semelhante ao dos dispositivos reais. Nessa perspectiva, o Mininet-WiFi apresenta-

se como uma plataforma para emulação de redes sem fio baseada no módulo mac80211_hwsim nativo do *kernel* do sistema operacional Linux [Fontes et al. 2015]. A seguir, será detalhada a importância da emulação e uma apresentação introdutória do Mininet-WiFi.

3.3.1. Experimentação simulada, emulada ou real?

Existem algumas opções para os pesquisadores testarem suas hipóteses e avaliarem a performance de suas soluções para redes de comunicação de múltiplos drones, as formas mais tradicionais de experimentação são: simulada, emulada e real [Wang et al. 2013]. Todas elas possuem vantagens e desvantagens, portanto, é importante que o pesquisador analise qual a melhor opção para sua pesquisa, ou ainda, quais alternativas ele dispõe para alcançar seus objetivos experimentais.

A experimentação simulada modela as operações e interações entre os dispositivos em um ou mais softwares. Isso permite maior flexibilidade, escalabilidade e controle a um custo muito reduzido. Além de entregar a capacidade de repetição e acessibilidade a outros usuários, que, na maioria dos casos, agregar valor a pesquisa, visto que outros pesquisadores poderão repetir seus experimentos e, também, usar esse ambiente para extrair novos resultados e comparar com anteriores. Contudo, a qualidade dos resultados está relacionada de forma estreita com proximidade entre a modelagem e a realidade, ou seja, quanto mais eficiente a modelagem, mais fidedignos são os resultados obtidos. São exemplos conhecidos de simuladores para redes de computadores o ns-2[2] e ns-3[3].

Diferente da anterior, a experimentação emulada tem como característica principal a utilização de softwares reais, como por exemplo, *kernels* e *drivers*. Isso viabiliza a obtenção de resultados mais próximos da realidade. Outro fator a ser considerado é, na grande maioria dos casos, a portabilidade de código para a experimentação real com a necessidade de poucos ajustes. Um exemplo de software para emulação de redes de computadores é o Mininet[4]. Uma característica interessante é a compatibilidade com ferramentas já desenvolvidas, por exemplo, *iperf* e *ssh*, o que agrega mais possibilidades ou pesquisador na construção e manipulação do cenário. Como na experimentação simulada, também permite maior capacidade de reprodutibilidade e repetibilidade. Destaca-se que soluções emuladas também fazem uso de simulação para conseguir construir seus cenários.

Por fim, a experimentação real é feita com dispositivos de hardware e software reais, normalmente, em um ambiente controlado. Isso implica na construção ou utilização de *testbeds*. A grande vantagem é gerar resultados muito próximos da realidade. Contudo, na maioria dos casos, demanda um custo financeiro elevado, alta complexidade para realização dos experimentos, além da baixa disponibilidade para outros usuários. Logo, as desafios impostos as capacidades de reprodutibilidade e repetibilidade da pesquisa são elevadas. Exemplos de *testbeds* são o Emulab[5] e PlanetLab[6].

[2]http://nsnam.sourceforge.net/wiki/index.php/Main_Page
[3]https://www.nsnam.org/
[4]http://mininet.org/
[5]https://www.emulab.net/portal/frontpage.php
[6]https://planetlab.cs.princeton.edu/

3.3.2. Mininet-WiFi

O Mininet-WiFi foi desenvolvido como uma extensão do Mininet para atuar com redes de comunicação sem fio, sejam elas baseadas no paradigma de Redes Definidas por Software (*Software-Defined Networking* - SDN) ou não. Apesar do termo WiFi fazer referência ao IEEE 802.11 e suas versões(a, b, g, n, ac, etc.), atualmente o emulador suporte diversas tecnologias e protocolos, entre eles IEEE 802.15.4 e LTE. Isso amplia e diversifica a construção de cenários, inclusive viabiliza ambientes de redes heterogêneas. Um exemplo da construção desse tipo de cenário pode ser observado no trabalho de [Selvaraju et al. 2021].

A instalação do Mininet-WiFi é simples, basta utilizar o script *install.sh* disponibilizado no repositório oficial do projeto no GitHub[7]. Para tal, é necessário fazer donwload ou clonar o reposiório para a estação onde deseja realizar a instalação. Além disso, existe a opção de fazer uso da máquina virtual pré-configurada disponível no mesmo repositório. Mais informações sobre instalação e detalhes sobre o Mininet-Wifi podem ser obtidos no livro[8], que possui capítulos gratuitos para comunidade, a página oficial do projeto e publicações que fizeram uso do emulador.

3.3.3. Emulando Redes Sem Fio no Mininet-WiFi

A emulação de cenários básicos de comunicação para redes sem fio no Mininet-WiFi não exige um alto grau de conhecimento prévio. É possível criar cenários através do terminal, com as topologias *single* e *linear*, por exemplo, ou através de scripts *python*, nesse formato é possível uma maior customização. Uma grande variedade de scripts estão disponibilizados no diretório de exemplos e devem ser utilizados como base para cenários mais complexos. A seguir será criado um cenário pelo terminal e alguns comandos serão apresentados. Parte dos comandos são nativos do Mininet-WiFi e outros são ferramentas disponíveis no ambiente Linux.

Prática 01: Criando um cenário básico pelo terminal

OBJETIVO: Criar cenários simples a partir do terminal, aprender alguns comandos do Mininet-WiFi e conhecer algumas ferramentas importantes para monitorar e manipular redes sem fio.

COMANDO: sudo mn –wifi

DESCRIÇÃO: Criar o cenário básico do Mininet-WiFi. Ele é composto por um controlador (c1), um access point(ap1) e duas estações (sta1 e sta2).

```
wifi@wifi-VirtualBox:~$ sudo mn --wifi
*** Adding stations:
sta1 sta2
*** Adding access points:
ap1
```

[7]https://github.com/intrig-unicamp/mininet-wifi

[8]https://mininet-wifi.github.io/book/

```
*** Configuring wifi nodes...
*** Creating network
*** Adding controller
*** Adding hosts:

*** Adding switches:

*** Adding links:
(sta1, ap1) (sta2, ap1)
*** Starting controller(s)
c0
*** Starting L2 nodes
ap1 ...
*** Starting CLI:
mininet-wifi>
```

COMANDO: nodes

DESCRIÇÃO: Listar os nós ativos no cenário em execução.

```
mininet-wifi> nodes
available nodes are:
ap1 c0 sta1 sta2
```

COMANDO: links

DESCRIÇÃO: Listar os links ativos no cenário em execução.

```
mininet-wifi> links
sta1-wlan0<->wifi (OK wifi)
sta2-wlan0<->wifi (OK wifi)
```

COMANDO: iw dev <interface> info

DESCRIÇÃO: Listar informações de uma interface de rede e seu estado. Para mais informações sobre o utilitário *iw* acesse sua documentação[9].

```
mininet-wifi> sta1 iw dev sta1-wlan0 info
Interface sta1-wlan0
        ifindex 46
        wdev 0x1800000002
        addr 02:00:00:00:00:00
        ssid my-ssid
        type managed
        wiphy 24
        channel 1 (2412 MHz), width: 20 MHz (no HT), center1: 2412 MHz
        txpower 14.00 dBm
```

[9] https://wireless.wiki.kernel.org/en/users/documentation/iw

COMANDO: iw dev <interface> link

DESCRIÇÃO: Listar informações do link de comunicação de uma interface de rede, caso esteja conectado.

```
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:02:00 (on sta1-wlan0)
        SSID: my-ssid
        freq: 2412
        RX: 341040 bytes (7802 packets)
        TX: 2353 bytes (27 packets)
        signal: -36 dBm
        tx bitrate: 5.5 MBit/s

        bss flags:        short-slot-time
        dtim period:        2
        beacon int:        100
```

COMANDO: iw dev <interface> scan

DESCRIÇÃO: Listar redes sem fio disponíveis para associação.

```
mininet-wifi> sta1 iw dev sta1-wlan0 scan
BSS 02:00:00:00:02:00(on sta1-wlan0) -- associated
        TSF: 1632158575581557 usec (18890d, 17:22:55)
        freq: 2412
        beacon interval: 100 TUs
        capability: ESS ShortSlotTime (0x0401)
        signal: -36.00 dBm
        last seen: 0 ms ago
        Information elements from Probe Response frame:
        SSID: my-ssid
        Supported rates: 1.0* 2.0* 5.5* 11.0* 6.0 9.0 12.0 18.0
        DS Parameter set: channel 1
        ERP: Barker_Preamble_Mode
        Extended supported rates: 24.0 36.0 48.0 54.0
        Supported operating classes:
                * current operating class: 81
        Extended capabilities:
                * Extended Channel Switching
                * SSID List
                * Operating Mode Notification
```

COMANDO: ping <node >

DESCRIÇÃO: Teste de conectividade camada 3. Faz uso do protocolo ICMP. Também calcula a variação entre os tempo de envio e de recebimento de cada pacote. Emite um relatório simples com estatísticas sobre o teste efetuado.

```
mininet-wifi> sta1 ping -c5 sta2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.125 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.136 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.142 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.141 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4075ms
rtt min/avg/max/mdev = 0.093/0.127/0.142/0.020 ms
```

COMANDO: iw dev <interface> disconnect

DESCRIÇÃO: Efetuar a desassociação da interface da rede sem fio.

```
mininet-wifi> sta1 iw dev sta1-wlan0 disconnect
mininet-wifi> sta1 iw dev sta1-wlan0 link
Not connected.
mininet-wifi> sta1 ping -c1 sta2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

COMANDO: iw dev <interface> connect <ssid>

DESCRIÇÃO: Efetuar a associação da interface a uma rede sem fio.

```
mininet-wifi> sta1 iw dev sta1-wlan0 connect my-ssid
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:02:00 (on sta1-wlan0)
        SSID: my-ssid
        freq: 2412
        RX: 5915 bytes (134 packets)
        TX: 349 bytes (4 packets)
        signal: -36 dBm
        tx bitrate: 1.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
mininet-wifi> sta1 ping -c1 sta2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.350 ms
```

```
--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.350/0.350/0.350/0.000 ms
```

COMANDO: xterm < node >

DESCRIÇÃO: Inicia um terminal emulado para um nó.

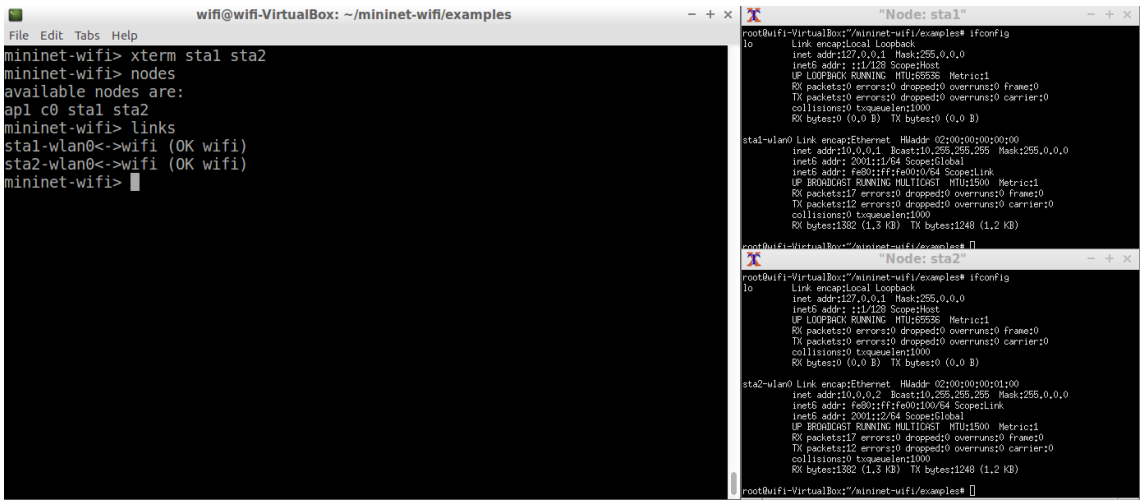

Figura 3.3. Múltiplos terminais com Xterm.

COMANDO: exit

DESCRIÇÃO: Encerra o cenário e o Mininet-WiFi.

```
mininet-wifi> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping switches/access points
ap1
*** Stopping nodes
sta1 sta2

*** Removing WiFi module and Configurations
*** Killing mac80211_hwsim

*** Done
completed in 959.645 seconds
wifi@wifi-VirtualBox:~$
```

Prática 02: Criando um cenário a partir de scripts

OBJETIVO: Criar cenários customizados a partir de scripts python. Considerando as tecnologias utilizadas em FANETs, o foco principal é construir redes ad hoc. Será utilizado como ponto de partida o arquivo *adhoc.py*[10] localizado no diretório *examples* do Mininet-WiFi. Recomenda-se a leitura detalhada do código fonte para melhor entendimento.

COMANDO: sudo python adhoc.py

DESCRIÇÃO: Executar o script adhoc.py.

```
/home/wifi/mininet-wifi/examples$ sudo python adhoc.py
*** Creating nodes
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Creating links
*** Starting network
sta1 sta2 sta3
*** Addressing...
*** Running CLI
*** Starting CLI:
mininet-wifi>
```

COMANDO: iw dev <interface> link

```
mininet-wifi> sta1 iw dev sta1-wlan0 link
Joined IBSS 02:ca:ff:ee:ba:01 (on sta1-wlan0)
        SSID: adhocNet
        freq: 2432
mininet-wifi> sta2 iw dev sta2-wlan0 link
Joined IBSS 02:ca:ff:ee:ba:01 (on sta2-wlan0)
        SSID: adhocNet
        freq: 2432
mininet-wifi> sta3 iw dev sta3-wlan0 link
Joined IBSS 02:ca:ff:ee:ba:01 (on sta3-wlan0)
        SSID: adhocNet
        freq: 2432
```

EXERCÍCIO 01: Visto que o enlace de comunicação ad hoc está criado, utilize os comandos já apresentados para realizar análises no cenário e verificar se há conectividade entre todos nós.

[10]https://github.com/intrig-unicamp/mininet-wifi/blob/master/examples/adhoc.py

RESPOSTA 01: O cenário é composto por três estações (sta1, sta2 e sta3). Existe o enlace de comunicação ad hoc com ssid *adhocNet*. Ao realizar testes de conectividade entre as três estações percebe-se que a sta1 não consegue enviar/receber pacotes ICMP gerados pelo utilitário *ping* que tenha como destino a sta3.

EXERCÍCIO 02: Feito o diagnóstico, qual a melhor alternativa para viabilizar conectividade entre todos os nós do cenário?

RESPOSTA 02: Para facilitar a busca por soluções, será inserido a linha de código $net.plotGraph(max_x = 100, max_y = 100)$ após a linha $net.configureWifiNodes()$. Para tal, finalize a emulação atual e executa o script *adhoc.py* novamente. Agora será exibido uma representação gráfica do cenário conforme Figura 3.4.

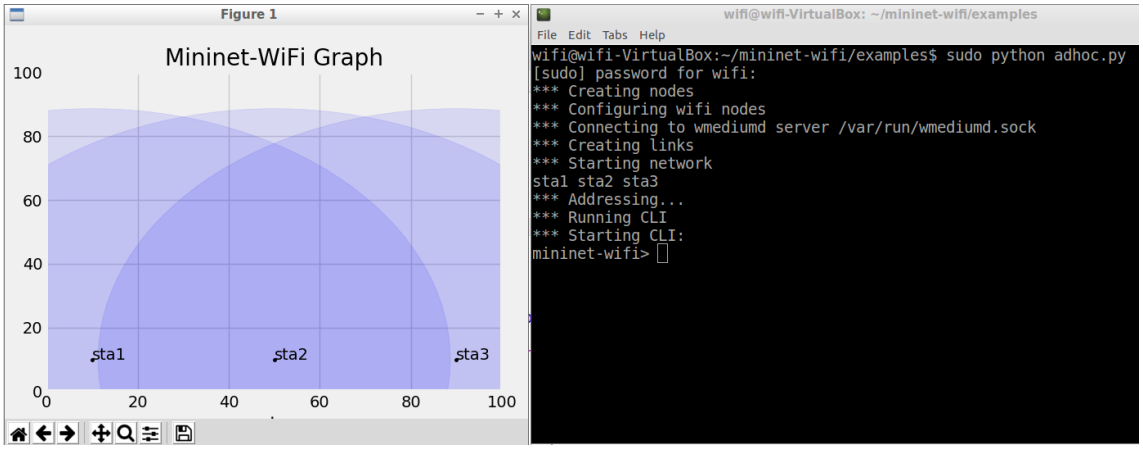

Figura 3.4. Execução do script adhoc.py e visualização do cenário no Mininet-WiFi.

SOLUÇÃO 01: É possível perceber que sta3 não está na área de cobertura da sta1. Apesar de sta2 ter em sua área de cobertura as duas estações, ela não é capaz de viabilizar o encaminhamento de pacotes. Dessa forma, uma alternativa para viabilizar a comunicação entre sta1 e sta3 é alterar a posição de uma das duas estações. A seguir a posição de sta3 será alterada para ingressar na área de cobertura da sta1.

COMANDO: py <node>.setPosition((’x,y,z’)

DESCRIÇÃO: Alterar o posição(x,y,z) de um nó. Para consultar a posição basta usar py <node>.position. É possível encontrar a distância entre dois nós através do comando *distance node1 node2*. Para mais informações sobre comandos no Mininet-WiFI acesse a página oficial no menu comandos[11].

[11] https://mininet-wifi.github.io/commands/

```
mininet-wifi> py sta3.position
[90.0, 10.0, 0.0]
mininet-wifi> py sta3.setPosition('80.0,10.0,0.0')
mininet-wifi> py sta3.position
[80.0, 10.0, 0.0]
mininet-wifi> sta1 ping -c5 sta3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=224 ms

#--- 10.0.0.3 ping statistics ---
5 packets transmitted, 1 received, 80% packet loss, time 4122ms
rtt min/avg/max/mdev = 224.629/224.629/224.629/0.000 ms
```

SOLUÇÃO 02: Conforme observado, sta2 é um elemento intermediário entre sta1 e sta3. Dessa forma, é possível implementar uma solução sem alterar a disposição geográfica dos nós, porém é preciso fazer um de um protocolo de roteamento. O próprio script adhoc.py dispõe de três opções de protocolos, são eles: OLSR, B.A.T.M.A.N e Babel. Não será feito um detalhamento do funcionamento dos protocolos. Durante esse capítulo serão utilizados apenas os dois primeiros.

INSTALAÇÃO: OLSR e B.A.T.M.A.N

DESCRIÇÃO: Para realizar a instalação dos protocolos OLSR e B.A.T.M.A.N, basta utilizar o script *install.sh* dentro do diretório util do Mininet-WiFi utilizando os parâmetro -O e -B, respectivamente.

COMANDO 1: /home/wifi/mininet-wifi$ sudo util/install.sh -O

COMANDO 2: /home/wifi/mininet-wifi$ sudo util/install.sh -B

Após a instalação dos procolos, é necessário apenas executar o script *adhoc.py* com o protocolo desejado como parâmetro. Por exemplo, olsrd ou batman_adv. Por questões de limitação de espaço, só será exibido a execução do OLSR.

```
/home/wifi/mininet-wifi/examples$ sudo python adhoc.py olsrd
*** Creating nodes
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Creating links
Starting olsrd in sta1-wlan0...
Starting olsrd in sta2-wlan0...
Starting olsrd in sta3-wlan0...
*** Starting network
```

```
sta1 sta2 sta3
*** Addressing...
*** Running CLI
*** Starting CLI:
mininet-wifi>
```

Executando o novo teste de conectividade.

```
mininet-wifi> sta1 ping -c5 sta3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=63 time=4.28 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=63 time=2.09 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=63 time=2.60 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=63 time=2.71 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=63 time=19.5 ms

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 2.094/6.245/19.535/6.685 ms
```

Observe a seguir a tabela de roteamento da sta1. Para datagramas que tenham como destino o ip 10.0.0.3 deve-se encaminhar para o gateway 10.0.0.2. Portanto, o protocolo OLSR está atualizando a tabela de roteamento da estação.

```
mininet-wifi> sta1 route
Kernel IP routing table
Destination Gateway  Genmask         Flags Metric Iface
10.0.0.0    *        255.0.0.0       U     0      sta1-wlan0
10.0.0.2    10.0.0.2 255.255.255.255 UGH   2      sta1-wlan0
10.0.0.3    10.0.0.2 255.255.255.255 UGH   2      sta1-wlan0
```

3.4. Emulação de Redes de Comunicação para Sistemas de Múltiplos Drones

A construção de um ambiente integrado para emulação de redes de comunicação de sistemas de múltiplos drones é uma demanda crescente entre os pesquisadores. Como elementos motivadores está a explosão de dispositivos capazes de trocar dados, dada a realidade cada vez mais presente da Internet das Coisas (*Internet of Things* - IoT), aliados a popularização dos drones, em especial aqueles com capacidade de realizar missões de forma autônoma. Isso promete revolucionar algumas áreas, entre elas as de logística e de vigilância, por exemplo.

A Figura 3.5 apresenta uma consulta a base de dados Scopus, realizada no dia 23 de setembro de 2021, com a palavra chave ”multi-uav”nos campos título, palavra-chave e resumo. Foi feito um recorte temporal dos últimos dez anos e mostrou o crescimento no número de publicações. Somados, chega-se a um total de 1.414 publicações. Destaque para o ano de 2020 com 304 documentos publicados.

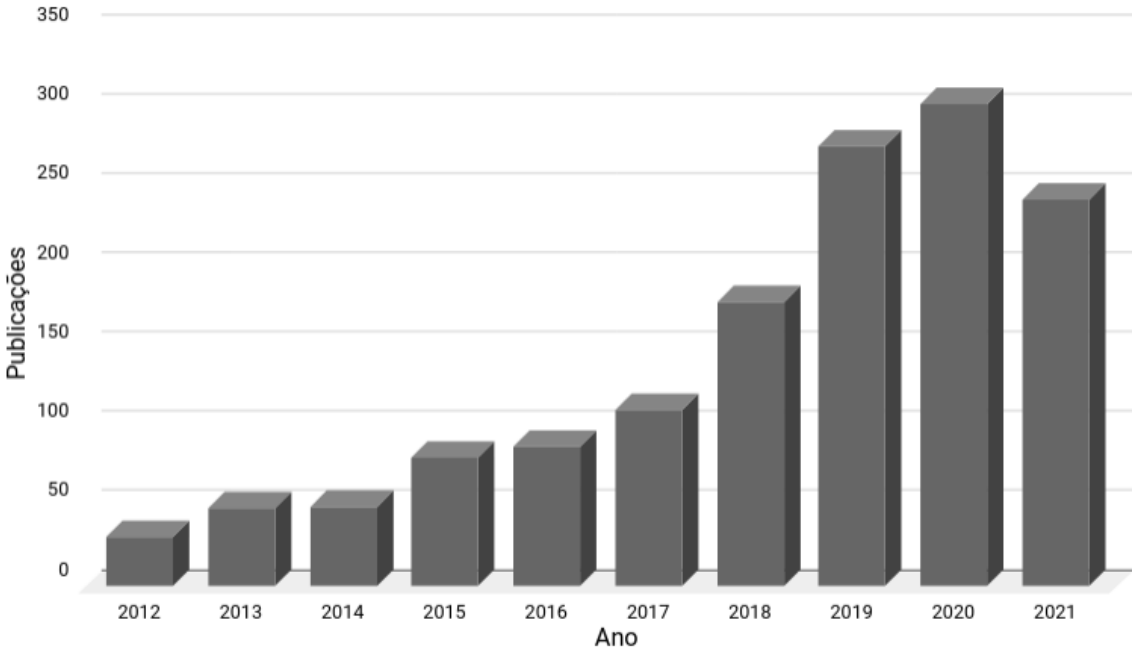

Figura 3.5. Execução do script adhoc.py e visualização do cenário no Mininet-WiFi.

Dentro desse contexto, validar pesquisas nessa temática, independente da estratégia de experimentação, ainda é um grande desafio. Seja pelo custo, hardware e/ou software altamente especializados e caros, ou pela complexidade de integrar diferentes sistemas, tecnologias e até hardware para conseguir um ambiente válido para avaliação de resultados.

Uma alternativa para esse ambiente é a integração do Mininet-WiFi com o simulador robótico CoppeliaSim. Conforme a página oficial do simulador, o CoppeliaSim pode ser usado para desenvolvimento rápido de algoritmos, simulações de automação em fábricas, prototipagem, robótica educacional, monitoramento remoto, verificação dupla de segurança através do gêmeo digital, entre outras. É um simulador de fácil instalação, multiplataforma e disponibiliza uma versão educacional.

3.4.1. Integração Mininet-WiFi e CoppeliaSim

Para realizar a integração entre o Mininet-WiFi e o CoppeliaSim será adotado o sistema operacional Linux. Como informado anteriormente, a página oficial do Mininet-WiFi disponibiliza uma máquia virtual Linux pré-configurada. Para fazer uso dessa máquina é necessário a instalação de um software capaz de executar máquinas virtuais, sugere-se o VirtualBox[12]. Após instalação, basta importar a máquina e iniciá-la.

A integração só é possível devido a disponibilidade de uma API fornecida pelo CoppeliaSim. A API remota permite que aplicações externas ao CoppeliaSim tenham a capacidade de trocas informações com o simulador. Ela é disponibilizada em algumas linguagens de programação, entre elas: C/C++, Java, Python, MatLab, Octave e Lua. Inclusive, Lua é a linguagem de programação nativa do CoppeliaSim. Mais detalhes

[12]https://www.virtualbox.org/

podem ser obtidos no manual do simulador[13].

O suporte a integração entre as plataformas já está disponível no diretório *mininet-wifi/examples/uav/* do Mininet-WiFi. A Figura 3.6 mostra o conteúdo do diretório. Os arquivos sim.py, simConst.py e remoteApi.so compõe a API remota para aplicações python em ambiente Linux. O arquivo simplesTest.py é um exemplo disponibilizado pela CoppeliaSim para verificação do funcionamento da API. O arquivo simulation.ttt é o cenário executado pelo CoppeliaSim, ele contém os drones e todas as informações do ambiente. O arquivo getNodePosition.py captura a posição dos drones no simulador. O arquivo setNodePosition atualiza a posição dos nós no Mininet-WiFI. Por fim, o arquivo uav.py faz a integração entre as plataformas utilizando os arquivos anteriores.

Figura 3.6. Arquivos do diretório *mininet-wifi/examples/uav/*.

Prática 03: Integrando Mininet-WiFI e CoppeliaSim

OBJETIVO: Realizar o setup do ambiente para integração entre o Mininet-WiFi e executar o primeiro cenário utilizando drones do CoopeliaSim. A Figura 3.7 mostra o cenário em execução.

COMANDO: ./examples/uav/install.sh

DESCRIÇÃO: O script install.sh detecta a distribuição Linux, faz o download do CoppeliaSim e descompacta o arquivo no diretório */home/wifi/mininet-wifi/examples/uav/*. De forma opcional, é possível acessar a página oficial do CoppeliaSim, realizar o download conforme a distribuição Linux utilizada e descompactá-lo no diretório *mininet-wifi/examples/uav/*.

COMANDO: sudo python examples/uav/uav.py

DESCRIÇÃO: Executa o script uav.py. O Mininet-WiFi e o Coppelia são iniciados. O cenário é composto por três drones, representados por estações no Mininet-WiFi e quadrirrotores no CoppeliaSim. É feito monitoramento contínuo das posições das aeronaves no CoppeliaSim para serem atualizadas no Mininet-WiFi. Formalmente, os

[13]https://www.coppeliarobotics.com/helpFiles/index.html

software não se comunicam diretamente, apesar da API remota, que é executada no arquivo SimplesText.py.

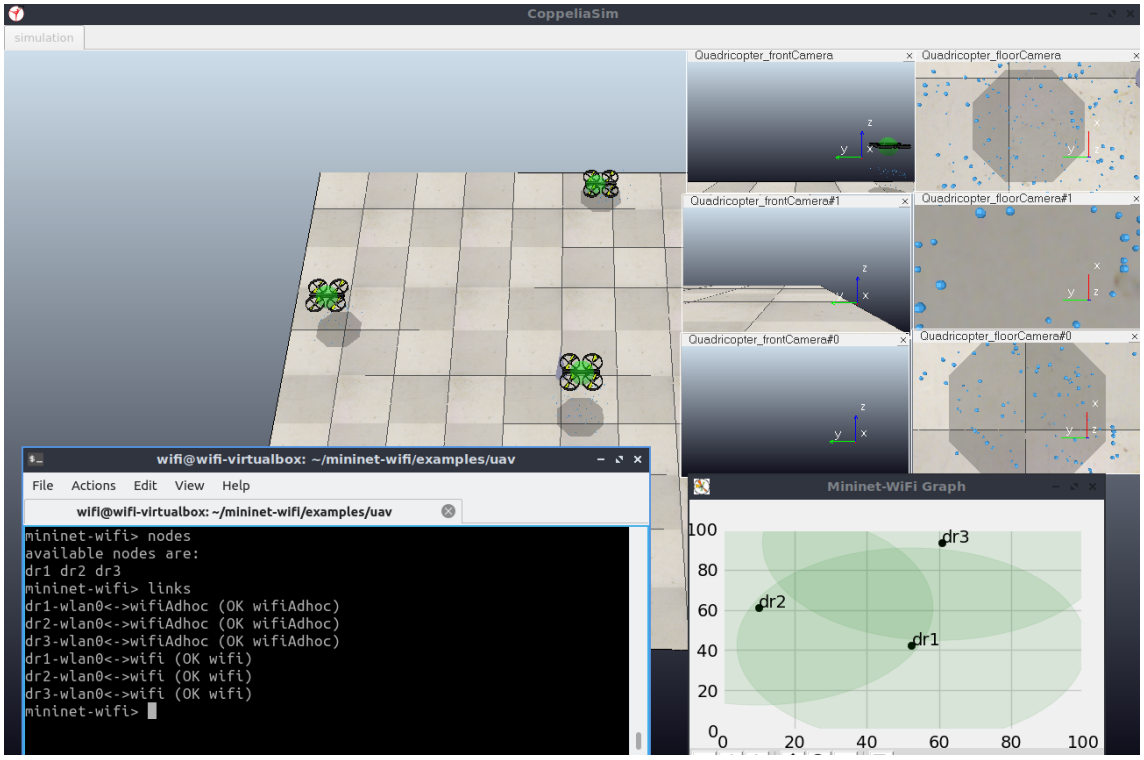

Figura 3.7. Cenário em execução.

Observe os nós ativos e os links. Os nós receberam ”dr”como nome seguido de um número identificador. As demais informações já foram apresentadas nas seções anteriores.

```
mininet-wifi> nodes
available nodes are:
dr1 dr2 dr3
mininet-wifi> links
dr1-wlan0<->wifiAdhoc (OK wifiAdhoc)
dr2-wlan0<->wifiAdhoc (OK wifiAdhoc)
dr3-wlan0<->wifiAdhoc (OK wifiAdhoc)
dr1-wlan0<->wifi (OK wifi)
dr2-wlan0<->wifi (OK wifi)
dr3-wlan0<->wifi (OK wifi)
```

A seguir um teste de conectividade com a ferramente ping entre os drones dr1 e dr2. Destaca-se novamente que toda a emulação relativa a comunicação ocorre no Mininet-WiFi.

```
mininet-wifi> dr1 ping -c5 dr2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
```

```
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=22.2 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.27 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=2.37 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=3.59 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=2.29 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 2.273/6.551/22.230/7.855 ms
```

Devido a movimentação das aeronaves, em algumas momentos da simulação os drones ficam sem conectividade de acordo com a capacidade da área de cobertura provida pela antena embarcada. Logo, garantir que as aeronaves permaneçam sempre na área de outra é um desafio abordado em muitas pesquisas.

```
mininet-wifi> dr1 ping -c5 dr2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4081ms
```

3.4.2. Entendendo o código fonte

A seguir será feito um breve detalhamento de parte do código fonte utilizado na construção do cenário. É importante entender sua estrutura para facilitar a customização de novos cenários.

ARQUIVO: uav.py

DESCRIÇÃO: Inserindo nós do tipo *Station* que serão a representação do drone no ambiente emulado.

```
info("*** Creating nodes\n")
    dr1 = net.addStation('dr1', mac='00:00:00:00:00:01',
                         ip='10.0.0.1/8', position='30,60,0')
    dr2 = net.addStation('dr2', mac='00:00:00:00:00:02',
                         ip='10.0.0.2/8', position='70,30,0')
    dr3 = net.addStation('dr3', mac='00:00:00:00:00:03',
                         ip='10.0.0.3/8', position='10,20,0')
```

DESCRIÇÃO: Inclusão dos links de comunicação do tipo adhoc. No cenário inicial foi utilizado o protocolo de roteamento batman_adv.

```
info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()
```

```
    net.addLink(dr1, cls=adhoc, intf='dr1-wlan0',
                ssid='adhocNet', proto='batman_adv',
                mode='g', channel=5, ht_cap='HT40+')

    net.addLink(dr2, cls=adhoc, intf='dr2-wlan0',
                ssid='adhocNet', proto='batman_adv',
                mode='g', channel=5, ht_cap='HT40+')

    net.addLink(dr3, cls=adhoc, intf='dr3-wlan0',
                ssid='adhocNet', proto='batman_adv',
                mode='g', channel=5, ht_cap='HT40+')
```

DESCRIÇÃO: Executando o script python setNodePosition.py passando como parâmetro os nomes dos drones(objetos stations).

```
info("*** Configure the node position\n")
    setNodePosition = 'python {}/setNodePosition.py '.format(path) +
                sta_drone_send + ' &'
    os.system(setNodePosition)
```

3.5. Performance de Redes de Comunicação para Sistemas de Múltiplos Drones

A avaliação da performance da rede sem fio responsável em viabilizar a comunicação entre as aeronaves (FANET) é fundamental para mitigar falhas na aplicação. Cada aplicação demanda por requisitos específicos de desempenho, portanto a infra-estrutura do sistema multi-VANT deve ser capaz de suprir as suas necessidades. Por exemplo, existem aplicações que demandam latência inferior a 1ms. Elas são classificas como aplicações de baixa latência (*low-latency*) e estão presentes em diversas áreas, desde saúde até transporte, tais como a realização de cirurgias remotas, utilização de carros autônomos, sistemas baseados em realidade aumentada, entre outras [Lema et al. 2017].

Dentro desse contexto, foram estabelecidas métricas de desempenham que são utilizadas para caracterizar a infraestrutura de comunicação. Elas devem ser consideradas desde a etapa de projeto de uma solução de conectividade, afinal o levantamento dos requisitos é uma das primeiras etapas de um projeto. Considerando os conceitos apresentados anteriormente, a configuração, aferição, análise e divulgação de resultados devem seguir padrões para garantir a reprodutibilidade da pesquisa e, além disso, a qualidade dos resultados obtidos [Rehman et al. 2010]. Como sugestão, recomenda-se a leitura da RFC 1242 *Benchmarking Terminology for Network Interconnection Devices* e da RFC 2544 *Benchmarking Methodology for Network Interconnect Devices*, além de suas atualizações.

3.5.1. Métricas de desempenho

Para efeitos de padronização, as métricas de desempenho apresentadas a seguir tiveram como referência a RFC 1242. As métricas apresentadas serão vazão (*throughput*), latência, *jitter* e perda de pacotes. Elas foram selecionadas por estarem alinhas com as ferramentas apresentadas na seção seguinte.

- **Vazão (*Throughput*)**: é a quantidade de dados recebidos corretamente em um intervalo de tempo, normalmente 1 segundo. Para mensurar o comportamento da vazão é importante utilizar tamanhos diferentes de quadros.
- **Latência**: para efeito de simplificação, a latência será considerada o tempo necessário para um pacote acessar o enlace de comunicação e ser completamente recebido no seu destino [Youm and Kim 2013].
- ***Jitter*:** é o intervalo de tempo entre transmissões e recebimentos de pacotes bem-sucedidas. Deve ser aferido de forma contínua. Ele é obtido a partir do desvio padrão da latência [Youm and Kim 2013]. Mais detalhes do cálculo do *Jitter* podem ser encontrados na RFC 3550.
- **Perda de Pacotes**: é uma taxa percentual de pacotes não recebidos ou com falhas.

3.5.2. Ferramentas para Avaliação de Performance

Atualmente, existem diversas ferramentas para avaliação de performance de redes de comunicação. Inclusive, elas estão integradas com sistemas responsáveis por monitorar a infraestrutura, isso inclui também dispositivos de interconexão, como roteadores, pontos de acesso e switches, como também servidores ou qualquer outros dispositivo conectado a rede. O objetivo é manter a alta disponibilidade dos serviços ofertados. Exemplos desse tipo de solução são o Netdata[14] e o Zabbix[15]. Contudo, este capítulo ficara restrito a duas ferramentas simples e bastante eficientes: ping e iperf. Elas são brevemente apresentadas a seguir.

- **Ping:** é uma ferramente utilizada, principalmente, para verificação de conectividade. Faz uso do protocolo ICMP e marcações de tempo para gerar relatórios e estatísticas. Dentre informações que seu relatório fornece, está o *Round Trip Time* (RTT). Nesse contexto, o RTT é a variação de tempo para a requisição ICMP ser respondida.
- **Iperf:** é uma ferramenta para aferição de métricas de performance de redes de comunicação baseadas na pilha de protocolos TCP/IP. Construída na arquitetura cliente/servidor, oferece opções de funcionamento para os protocolos TCP e UDP. Dentre as métricas disponíveis estão: largura de banda, perda de pacotes e jitter. A versão 3 contém mais funcionalidade que seu antecessor. Mais informações estão disponíveis na página oficial da ferramenta[16].

[14]https://www.netdata.cloud/
[15]https://www.zabbix.com/
[16]https://iperf.fr/

Prática 04: Utilizando Ferramentas para Avaliação de Performance de Rede

OBJETIVO: Utilizar as ferramentas apresentadas para aferir métricas de desempenho da rede. A partir disso, realizar alterações de configuração no cenário para efeitos de comparação com resultados obtidos anteriormente.

COMANDO: iperf -s

DESCRIÇÃO: Executar a ferramenta iperf no formato servidor TCP. Após isso, utilize o parâmetro -u para executá-la com o protocolo UDP.

```
root@elefante:~/mininet-wifi/examples/uav# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 45098
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.9 sec  6.38 MBytes  4.93 Mbits/sec
```

COMANDO: iperf -c <ip do servidor>

DESCRIÇÃO: Executar a ferramenta iperf no formato cliente TCP.

```
root@elefante:~/mininet-wifi/examples/uav# iperf -c 10.0.0.1
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.2 port 45098 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.4 sec  6.38 MBytes  5.16 Mbits/sec
```

EXERCÍCIO 03: Como extrair apenas o tempo de RTT médio a partir do relatório emitido pela ferramenta ping?

SOLUÇÃO: Utilizar os comandos tail e cut disponíveis no Linux para tratar a saída. Feito isso, é possível direcionar a saída para um arquivo e plotar gráficos.

```
ping -O -D -c10 10.0.0.1 | tail -1 | cut -d' ' -f4 | cut -d'/' -f2
```

EXERCÍCIO 04: É possível gerar arquivos ”csv”a partir dos relatórios emitidos pela ferramenta iperf?

SOLUÇÃO: Dentre os parâmetros disponíveis para a ferramenta iperf, está opção -y. Ela omite o relatório padrão da ferramenta e gera uma saída ”csv”. A disposição das colunas segue a seguinte sequência, quando emitido pelo servidor UDP: data-hora, ip-local, porta-local, ip-cliente, porta-cliente, id, tempo, total dados transmitidos, bandwith, jitter, pacotes perdidos, pacotes transmitidos, taxa de erro. Diante disso, para obter o jitter, pacotes perdidos e enviados pode-se utilizar o comando a seguir

```
iperf -u -s -yc | cut -d, -f10-12
```

3.6. Estudos de Caso - Inspeção em situações de emergência: incêndios

Devido a alta flexibilidade e rápida implantação dos sistemas multi-VANT, é possível utilizá-los em missões de preservação da vida. Em incêndios, principalmente, florestais, os quais ocorrem em grandes áreas, as aeronaves têm a capacidade de fornecer diversas informações, tais como: direção e intensidade do vento, temperatura, poluentes dispersos no ar, entre outras. Além disso, eles podem fornecer imagens aéreas da região permitindo aos operadores uma melhor visualização da situação. Inclusive, para localizar seres humanos, ou até animais, em situação de risco.

A Figura 3.8 ilustra uma situação de incêndio onde três seres humanos estão em risco. As aeronaves atuam de forma cooperativa para cobrir uma maior área de busca, desse modo é fundamental que eles possam trocas informações entre si. Observa-se que foi criada uma rede ad hoc para viabilizar a rápida implantação do sistema. Um protocolo de roteamento também é utilizado para habilitar a comunicação entre os drones que estão nas extremidade da topologia.

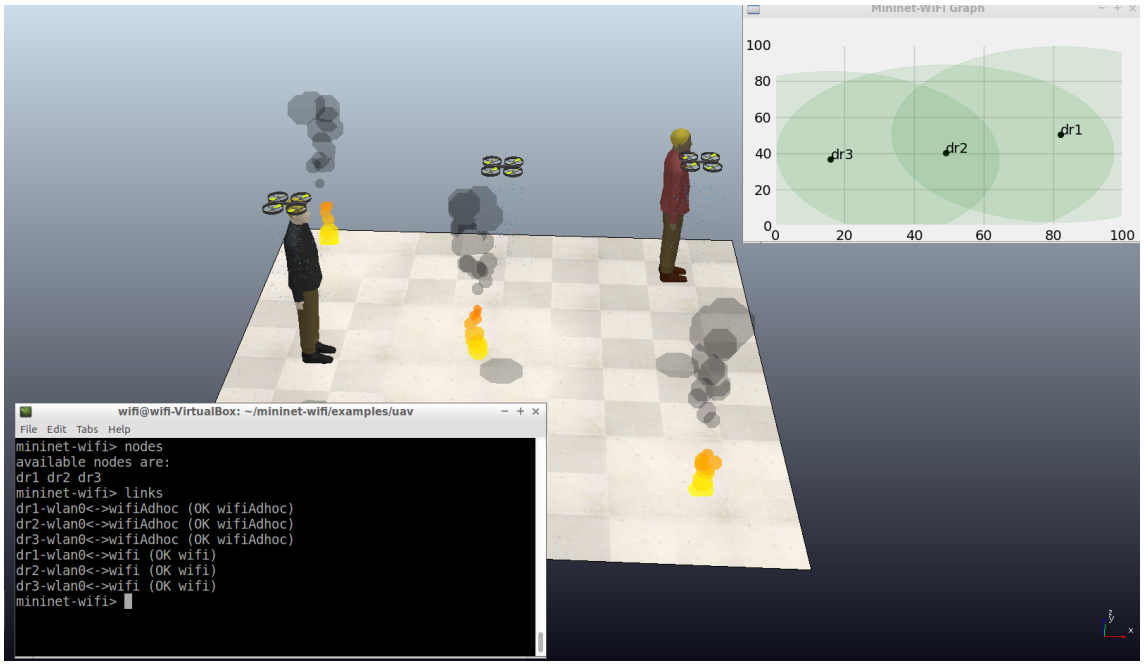

Figura 3.8. Cenário em execução.

Prática 05: Análise da rede de comunicação de um sistema multi-VANT

EXERCÍCIO 05: Faça o estudo da rede de comunicação do sistema multi-UAV. Utilize os comandos e ferramentas apresentados para comparar com o cenário apresentado anteriormente. Observe que no cenário de inspeção a navegação dos drones é controlada por um planejador de caminho.

3.7. Conclusão

Este capítulo apresentou uma estratégia para emular redes de comunicação para sistemas de múltiplos drones. Para tal, foram utilizados o emulador de redes sem fio Mininet-WiFi e o simulador CoppeliaSim. O Mininet-WiFi é responsável por emular o comportamento da rede de comunicação, enquanto o CoppeliaSim a movimentação das aeronaves, além do ambiente virtual que permite a visualização das aeronaves no cenário em execução.

Dentro desse contexto, percebe-se que aplicações que fazem uso de múltiplos drones atuando de forma cooperativa representam uma alternativa interessante para superar alguns problemas, principalmente quando relacionados a missões em locais de alto risco ou de difícil acesso. Contudo, é importante investigar soluções de conectividade para esses sistemas. Para tal, é fundamental apropriar-se de estratégias para construção de cenários e avaliação de performance conforme a aplicação que o sistema está inserido.

O estudo nessa temática e outras linhas aliadas a ela, como, por exemplo, Internet das Coisas, cresceu bastante nos últimos cinco anos. Entende-se que ainda existe um *gap* para poder atender as atuais demandas oriundas da Indústria 4.0. Outros avanços relevantes voltados para a tecnologia embarcada e para inteligência artificial vão posicionar os drones como elementos protagonistas nos próximos anos. Portanto, os sistemas de múltiplos drones devem seguir em forte desenvolvimento e expandindo ainda mais suas capacidades e aplicações.

Currículo dos autores

Diego da Silva Pereira possui graduação em Redes de Computadores pelo Instituto Federal do Rio Grande do Norte (IFRN), Mestrado em Ciência da Computação pela Universidade Estadual do Rio Grande do Norte (UERN) e está em doutoramento no Programa de Pós-graduação em Engenharia Elétrica e de Computação (PPgEEC) pela UFRN. Atualmente é professor no Instituto Federal do Rio Grande do Norte com pesquisas na área de redes de comunicação sem fio aplicadas à sistemas robóticos autônomos e aeroespaciais.

Luís Bruno Pereira do Nascimento possui Bacharelado em Ciência da Computação pela Universidade Estadual do Piauí (UESPI), Mestrado em Engenharia Elétrica e de Computação pela Universidade Federal do Ceará (UFC) e Doutorado em Engenharia Elétrica e de Computação (PPgEEC) pela Universidade Federal do Rio Grande do Norte (UFRN). Atualmente é professor no Instituto Federal do Rio Grande do Norte (IFRN). Suas áreas de interesse incluem robótica, otimização e aprendizado de máquina.

Vitor Gaboardi dos Santos possui graduação em Engenharia Elétrica na Universidade Federal do Rio Grande do Norte (UFRN) com período sanduíche na University of Kansas (USA) e Mestrado em Engenharia Mecatrônica pela UFRN. Atualmente, é professor substituto no Instituto Federal do Rio Grande do Norte (IFRN). Suas áreas de interesse incluem redes neurais, processamento de imagens e visão computacional.

Pablo Javier Alsina possui graduação em Engenharia Elétrica (1987), mestrado na área de controle de motores de indução (1991) e doutorado em Engenharia Elétrica com tema em controle de manipuladores robóticos (1996), pela Universidade Federal da Paraíba. Atualmente é professor titular do Departamento de Engenharia de Computação e Automação (DCA). É professor nos Programas de Pós-Graduação em Engenharia Mecatrônica (PPGEM) e em Engenharia Elétrica e de Computação (PPgEEC) da Universidade Federal do Rio Grande do Norte (UFRN), onde é chefe do Laboratório de Robótica. Desenvolve pesquisas em robótica, nas áreas de controle, planejamento e percepção robótica, com aplicações em robótica assistiva, robótica móvel e Veículos Aéreos Não Tripulados.

Referências

[Bai et al. 2021] Bai, C., Yan, P., Yu, X., and Guo, J. (2021). Learning-based resilience guarantee for multi-uav collaborative qos management. *Pattern Recognition*, page 108166.

[Chriki et al. 2019] Chriki, A., Touati, H., Snoussi, H., and Kamoun, F. (2019). Fanet: Communication, mobility models and security issues. *Computer Networks*, 163:106877.

[Fontes et al. 2015] Fontes, R. R., Afzal, S., Brito, S. H., Santos, M. A., and Rothenberg, C. E. (2015). Mininet-wifi: Emulating software-defined wireless networks. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 384–389. IEEE.

[Fu et al. 2019] Fu, Z., Mao, Y., He, D., Yu, J., and Xie, G. (2019). Secure multi-uav collaborative task allocation. *IEEE Access*, 7:35579–35587.

[Hong et al. 2021] Hong, Y., Jung, S., Kim, S., and Cha, J. (2021). Autonomous mission of multi-uav for optimal area coverage. *Sensors*, 21(7):2482.

[Lema et al. 2017] Lema, M. A., Laya, A., Mahmoodi, T., Cuevas, M., Sachs, J., Markendahl, J., and Dohler, M. (2017). Business case and technology analysis for 5g low latency applications. *IEEE Access*, 5:5917–5935.

[Pereira et al. 2020] Pereira, D. S., De Morais, M. R., Nascimento, L. B., Alsina, P. J., Santos, V. G., Fernandes, D. H., and Silva, M. R. (2020). Zigbee protocol-based communication network for multi-unmanned aerial vehicle networks. *IEEE Access*, 8:57762–57771.

[Rehman et al. 2010] Rehman, S. U., Turletti, T., and Dabbous, W. (2010). Benchmarking in wireless networks.

[Rooban et al. 2021] Rooban, S., Suraj, S. D., Vali, S. B., and Dhanush, N. (2021). Coppeliasim: Adaptable modular robot and its different locomotions simulation framework. *Materials Today: Proceedings*.

[Santos et al. 2019] Santos, V. G., Pereira, D. S., Alsina, P., Fernandes, D. H., Nascimento, L. B., Leite, D. L., Morais, M. R., Silva, M. R., and Souza, E. S. (2019). Multi-uav system architecture for environmental protection area monitoring. In *Proc. Anais do Simpsio Brasileiro de Automao Inteligente*, pages 1–6.

[Selvaraju et al. 2021] Selvaraju, S. P., Balador, A., Fotouhi, H., Vahabi, M., and Bjorkman, M. (2021). Network management in heterogeneous iot networks. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1581–1586. IEEE.

[Silva et al. 2019] Silva, M. R., Souza, E. S., Alsina, P. J., Leite, D. L., Morais, M. R., Pereira, D. S., Nascimento, L. B., Medeiros, A. A., Junior, F. H. C., Nogueira, M. B., et al. (2019). Performance evaluation of multi-uav network applied to scanning rocket impact area. *Sensors*, 19(22):4895.

[Wang et al. 2013] Wang, S.-Y., Chou, C.-L., and Yang, C.-M. (2013). Estinet openflow network simulator and emulator. *IEEE Communications Magazine*, 51(9):110–117.

[Wang et al. 2021] Wang, X., Yang, L. T., Meng, D., Dong, M., Ota, K., and Wang, H. (2021). Multi-uav cooperative localization for marine targets based on weighted subspace fitting in sagin environment. *IEEE Internet of Things Journal*.

[Youm and Kim 2013] Youm, S. and Kim, E.-J. (2013). Latency and jitter analysis for ieee 802.11 e wireless lans. *Journal of Applied Mathematics*, 2013.