Lab 2 : Compreendendo o Funcionamento das Redes Sem Fio com o Mininet-WiFi

Objetivos

Esta atividade prática introduz o emulador de redes Mininet-WiFi. Os objetivos desta atividade incluem:
• Compreender (em alto nível) o funcionamento do emulador de redes Mininet-WiFi.
• Explorar as features do emulador Mininet-WiFi
• Observar as diferenças entre o Mininet e o Mininet-WiFi

Preparação e Configuração

Neste Laboratório, será utilizado o mesmo ambiente dos laboratórios anteriores, porém os comandos seguintes para atualização dos repositórios devem ser executados.

Para garantir que a máquina virtual possua a versão mais recente do Mininet-WiFi, utilize o comando git pull para atualizar o código-fonte, conforme abaixo.

```
cd /home/wifi/mininet-wifi
sudo git pull
```

Agora para garantir que o repositório de atividades do Laboratório está na versão mais recente utilize o comando git clone, conforme abaixo (lembrando de exclui-lo caso ele já exista, como feito nos laboratórios anteriores).

```
cd /home/wifi/
git clone https://github.com/intrig-unicamp/EA080-2S2021.git
```

Antes de iniciar o laboratório, é necessário executar um *clean up* com o seguinte comando:

```
sudo mn -c
```

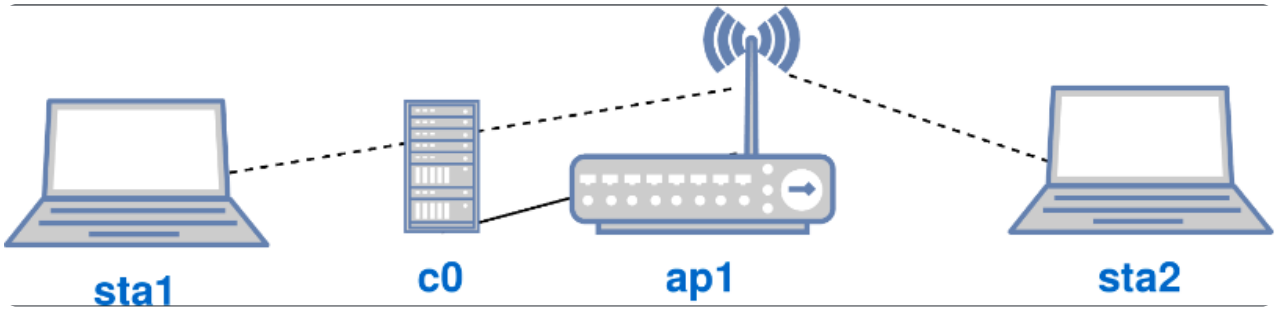
> Atenção: Durante as atividades práticas erros comuns ou pequenos bugs podem acontecer. A Seção FAQ apresenta alguns destes erros e o passo a passo para sua solução.

Atividades Práticas

Atividade 1: Primeiros passos

O comando abaixo permite criar uma rede simples, com 2 estações e um ponto de acesso:

```
sudo mn --wifi
```



O comando **help** traz uma lista de comandos disponíveis:

```
mininet-wifi> help
  Documented commands (type help <topic>):
  ========================================
  EOF      exit  iperf    nodes       pingpair     py     start  x
  distance gterm iperfudp noecho      pingpairfull quit   stop   xterm
  dpctl    help  links    pingall     ports        sh     switch
  dump     intfs net      pingallfull px           source time
```

Experimente utilizar o comando nodes para identificar os nós que fazem parte da rede.

```
mininet-wifi> nodes
```

Agora experimente os comandos abaixo. Eles listam as principais informações de cada nó e testar a comunicação entre eles.

```
mininet-wifi> sta1 iwconfig
mininet-wifi> sta2 iwconfig
mininet-wifi> sta1 ping sta2
```

Alternativamente, é possível acessar informações do nó com:

```
mininet-wifi> sta1 iw dev sta1-wlan0 info
```

Agora, desconecte sta1 e confirme a desconexão:

```
mininet-wifi> sta1 iw dev sta1-wlan0 disconnect
mininet-wifi> sta1 iwconfig
mininet-wifi> sta1 ping sta2
```

Agora, conecte sta1 novamente e teste a comunicação entre as estações:

```
mininet-wifi> sta1 iw dev sta1-wlan0 connect my-ssid
mininet-wifi> sta1 iwconfig
mininet-wifi> sta1 ping sta2
```

> P: Qual é o atraso observado entre sta1 e sta2? Houve perda de pacotes no canal? Justifique suas respostas de forma objetiva.

Você pode, teoricamente, utilizar na CLI do Mininet-WiFi qualquer comando disponível no Linux, incluindo utilitários como iw, iwconfig, etc. Porém, como trata-se de um ambiente emulado e nele podem estar múltiplos nós, você precisa identificar quem é o nó que será responsável por um dado comando (por exemplo, sta1 iwconfig).

> P: Use a ferramenta iperf para avaliar a banda disponível (Mbps) entre sta1 e sta2.

Atividade 2: Acessando informações dos nós e o ambiente

Se estiver dentro da CLI do Mininet-WiFi, vamos sair com exit.

```
mininet-wifi> exit
```

Agora, abra a mesma topologia acrescentando dois novos parâmetros: position e wmediumd. O parâmetro position vai definir posições iniciais para os nós; e o parâmetro wmediumd irá habilitar o wmediumd, um simulador do link wireless.

```
sudo mn --wifi --position --link=wmediumd
```

Tente utiliazar o comando "distance":

```
 mininet-wifi> distance sta1 sta2
```

Outros recursos disponíveis no Mininet-WiFi podem facilitar a interação com os nós. Por exemplo, o comando params permite identificar diversas informações do nó, como endereço ip, modo e frequência de funcionamento, dentre outros.

```
mininet-wifi> py sta1.params
```

É possível também filtrar informações específicas. No exemplo abaixo, é permite visualizar o IP:

```
mininet-wifi> py sta1.params['ip']
```

Ou frequência:

```
mininet-wifi> py sta1.params['freq']
```

Além dos atributos em params, podemos checar outras informações do nó, como portas:

```
mininet-wifi> py sta1.ports
```

Ou checar interfaces wireless:

```
mininet-wifi> py sta1.wintfs
```

E também checar posição:

```
mininet-wifi> py sta1.position
```

> P: Identifique a posição dos nós sta1 e sta2.

> Atenção: só será possível verificar a posição do nó se ela estiver sido definida previamente no código ou com a adição do argumento −−position.

Através da CLI do Mininet-WiFi o usuário também pode definir uma nova posição, por exemplo:

```
mininet-wifi> py sta1.setPosition('10,20,0')
```

E finalmente verificar a nova distância entre dois nós, como em:

```
mininet-wifi> distance sta1 sta2
```

> P: Investigue a lista de parâmetros de sta1 (py sta1.params). Explique resumidamente qual informação do nó sta1 cada item da lista representa.

> Observação: Por exemplo: 'ip': Lista o IP atual atribuído à interface wireless de STA1

Obs: É possível também analizar as interfaces do access point

```
mininet-wifi> ap1 ip link
```

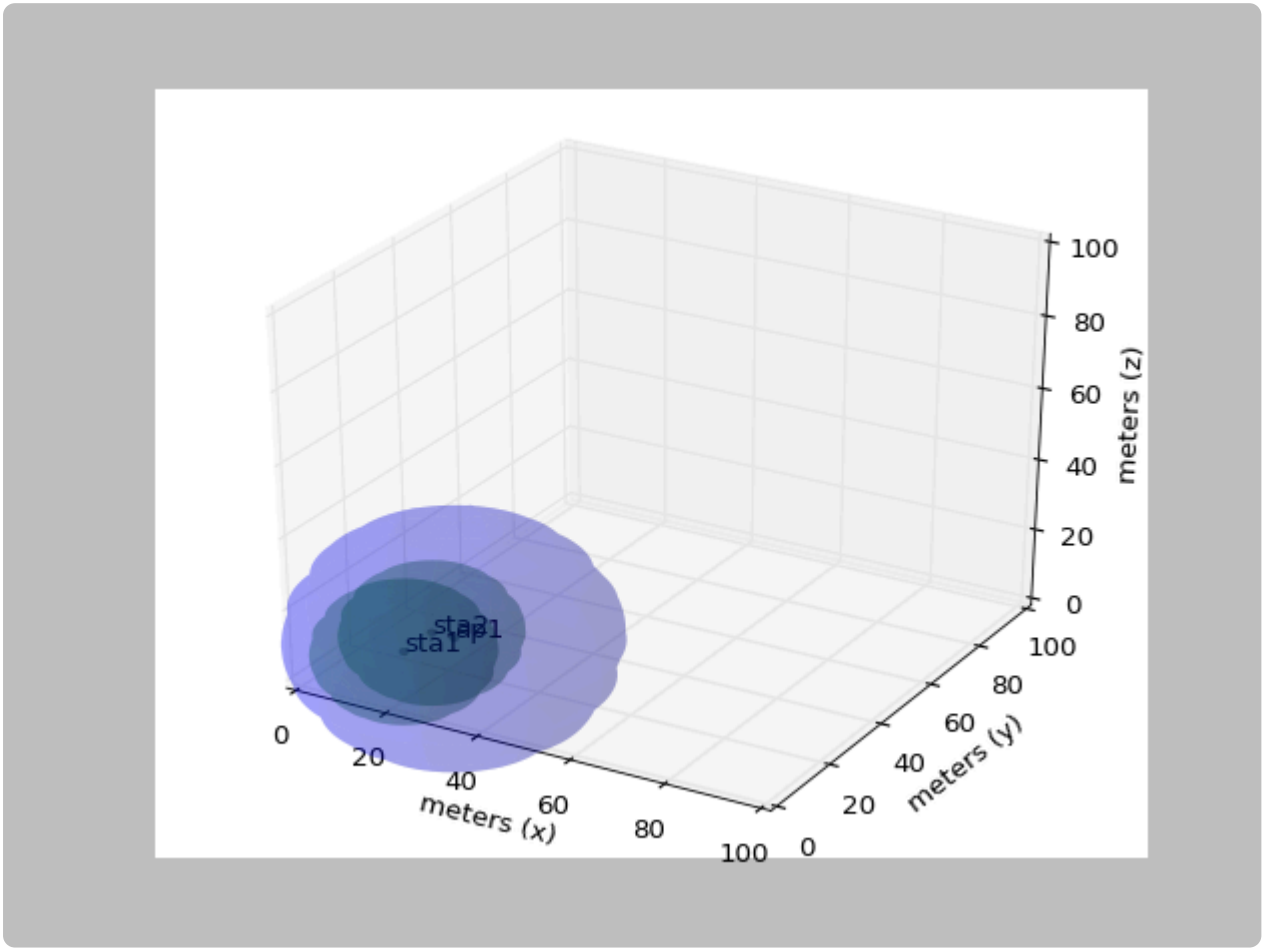
Atividade 3: Visualizando gráficos 2D e 3D

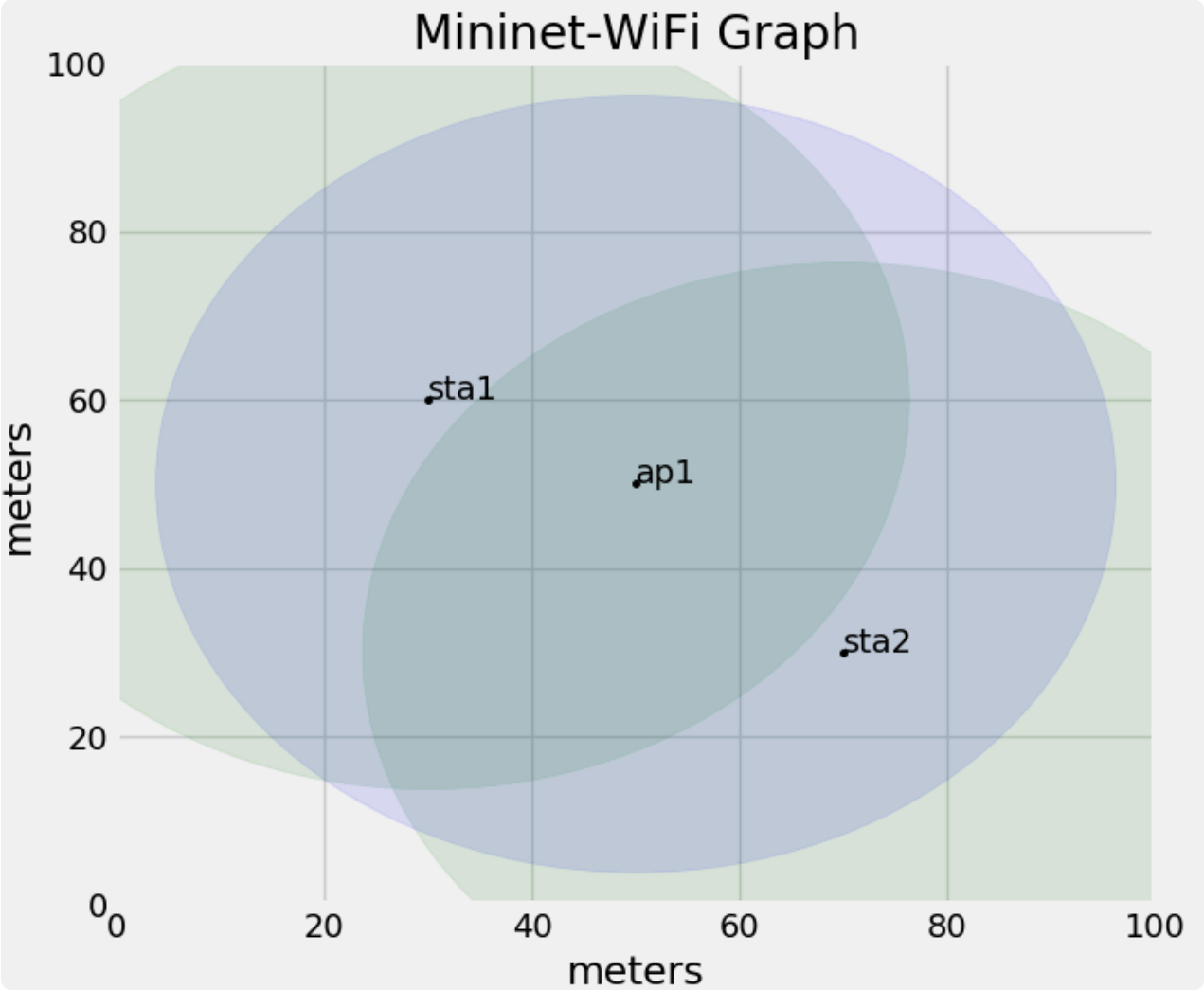
O Mininet-WiFi permite ao usuário executar topologias 2D e 3D. Em sua totalidade, os códigos disponíveis em `/mininet-wifi/examples` geram grafos em 3D. Caso o usuário deseje gerar gráficos em 2D, basta fazer uma simples alteração no código. Tomamos como exemplo o arquivo localizado em `/mininet-wifi/examples/position.py` Este arquivo contêm a seguinte linha (utilize um editor de texto, como o nano, para investigar o arquivo):

```
net.plotGraph(max_x=100, max_y=100, max_z=100)
```

Como pode ser visto acima, apenas os eixos x, y e z foram definidos e o resultado do gráfico será algo similar ao apresentado na primeira figura abaixo. Logo, para gerar gráficos em 2D basta remover um dos eixos (por exemplo o eixo z), que resultará em algo similar ao apresentado na segunda figura abaixo:

```
net.plotGraph(max_x=100, max_y=100)
```





Os gráficos gerados pelo Mininet-WiFi são suportados pelo matplotlib, uma biblioteca disponível na linguagem de programação Python que possibilita a criação de gráficos. Devido a uma limitação no matplotlib, todos os eixos precisam ser definidos com valores iguais.

> P: Plote um gráfico em duas e outro em três dimensões.

Atividade 4: Wireshark: Análise de quadros IEEE 802.11

Leia o Apêndice Wireshark e a captura de pacotes 802.11 que se encontra no final deste roteiro (após todas as atividades).

Capture pacotes em um enlace Ethernet com fio (IEEE 802.3) e depois em um enlace sem fio (IEEE 802.11) como feito na Atividade 1 (usando ping, por exemplo).

> P: Explique as principais diferenças e funções em relação ao número de endereços MAC contidos nos quadros do tipo 802.11 e 802.3.

Atividade 5: Modelos de Propagação

Sendo o Mininet-WiFi um emulador, ele precisa utilizar de modelos matemáticos para representar com maior fidelidade possível o comportamento do meio sem fio. Este comportamento é definido através de modelos matemáticos, que podem diferenciar uns dos outros acerca dos atributos suportados. Por exemplo, um modelo pode considerar a interferência entre os sinais provenientes de diferentes dispositivos sem fio, enquanto outros não.

Atualmente, o Mininet-WiFi suporta os seguintes modelos de propagação:

1. Friss Propagation Loss Model (model="friss")
2. Two Ray Ground Propagation Loss Model (model="twoRayGround")
3. Log Distance Propagation Loss Model (model="logDistance")
4. Log Normal Shadowing Propagation Loss Model (model="logNormalShadowing")
5. Internal Telecommunication Union Propagation Loss Model (model="ITU")

Para responder as questões das Atividades 6,7 e 8 (Extra) explore e execute os scripts disponíveis no diretório /lab3 do repositório EA080-2S2021. Caso esteja utilizando a máquina virtual o caminho para o diretório é:

```
/home/wifi/EA080-2S2021/lab3/
```

Para esta atividade utilizaremos o script `propagation_model.py` . Considere também utilizar comandos como iwconfig e iw (por exemplo: sta1 iwconfig) no terminal do Mininet-WiFi.

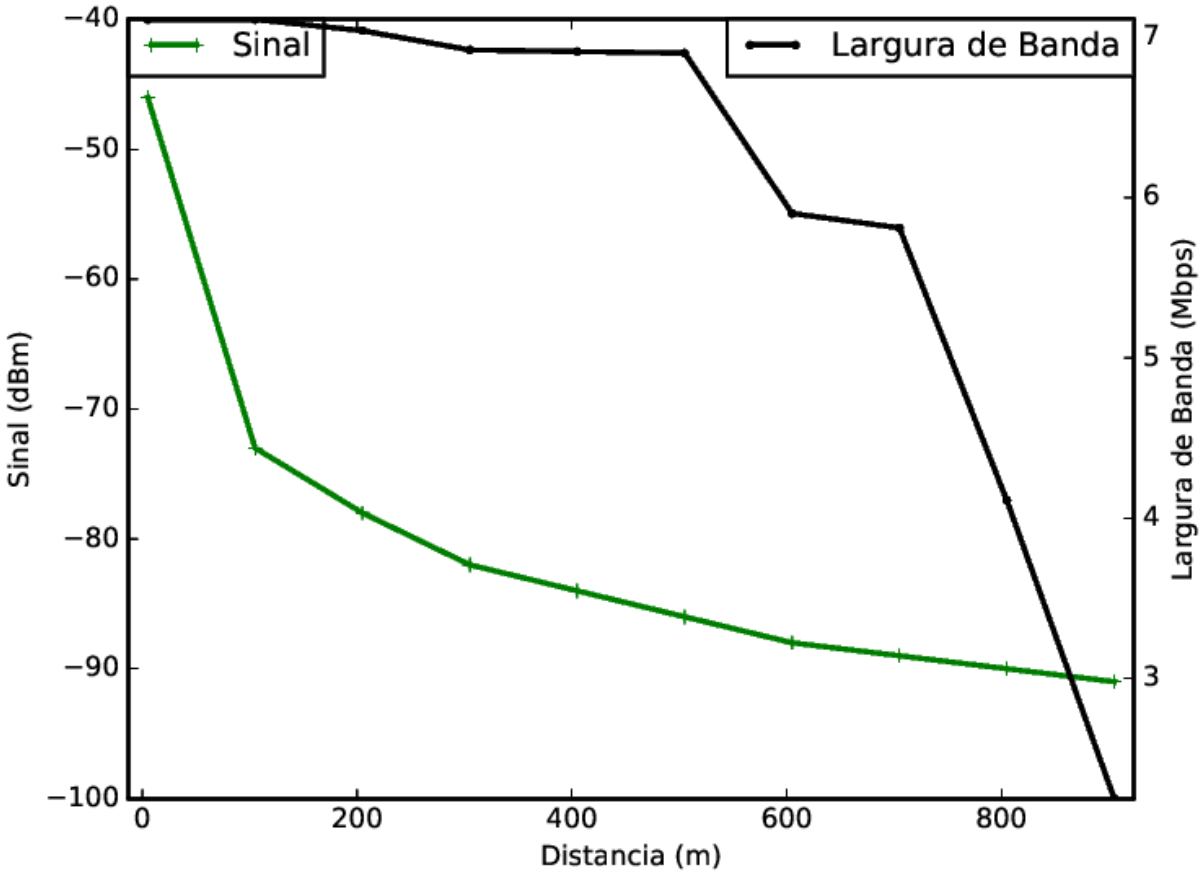
> Atenção: Caso o caminho especificado não exista ou o nome do arquivo esteja com extensão .txt, certifique-se de que você atualizou o repositório do laboratório com o comando git pull.

> P: Um dos nós parece estar incomunicável. Qual é ele e por qual motivo o nó está incomunicável? Ele encontra-se associado ao AP1? Tente responder de forma técnica.
>
> P: O script já vem pré-configurado com um modelo de propagação. Identifique-o.
>
> P: Identifique e descreva o nível de sinal percebido por sta1. Depois substitua o modelo de propagação do script por um outro suportado pelo Mininet-WiFi e então configure esse modelo no script.
>
> P: Há diferenças entre o nível de sinal antes e após a mudança do modelo de propagação? Justifique o resultado obtido.

Atividade 6: Relação Distância x Largura de Banda/RSS

Alguma das vantagens de utilizar modelos de propagação é o cálculo do nível de sinal percebido por uma estação a partir de alguns atributos, como a distância. A distância, por sua vez, impacta diretamente na transmissão de dados, uma vez que quanto menor o nível de sinal percebido, menor será a taxa de transmissão. O nível de sinal para redes WiFi normalmente se encontra entre 0 e -100 dBm. Quanto menor o valor, pior será o nível de sinal percebido.

A figura abaixo ilustra como exemplo uma possível relação entre Distância e Largura de Banda e RSSI a partir de um modelo de propagação aleatoriamente escolhido. Os valores para largura de banda apresentados na figura abaixo foram gerados a partir da saída da ferramenta Iperf que permite mensurar a largura de banda entre origem e destino.



P: Com o script propagation_model.py e utilizando Iperf, monte um gráfico ou tabela que relacione a largura de banda entre dois nós (sta) com a distância x entre eles, assim como exemplificado na figura. Monte também um gráfico ou tabela que relacione o nível de sinal à distância.

Observação: A escolha do modelo de propagação e sua configuração é livre. Não esqueça de destacar, para cada distância, o nível de sinal (RSSI) e largura de banda (Mbps) observados.

Atenção: O Mininet-WiFi possui um threshold, chamado de CCA Threshold (Clear Channel Assessment), que corta a comunicação quando o nível de sinal é inferior a -92 dBm.

Atividade 7: Mobilidade

O Mininet-WiFi também suporta modelos de mobilidade. Além dos modelos de mobilidade, uma outra opção disponível para o usuário é criar a mobilidade configurando a posição inicial e final do nó e também o tempo que o nó levará para percorrer os 2 pontos. O programa handover.py (http://handover.py), disponível no diretório, `/home/wifi/EA080-2S2021/lab3` executa esta ação:

```
#!/usr/bin/python

'Example for Handover'

from mininet.node import Controller
from mininet.log import setLogLevel, info
from mn_wifi.cli import CLI
from mn_wifi.net import Mininet_wifi


def topology():
    "Create a network."
    net = Mininet_wifi(controller=Controller)

    info("*** Creating nodes\n")
    sta1 = net.addStation('sta1', mac='00:00:00:00:00:02', ip='10.0.0.2/8',
                          range=20)
    sta2 = net.addStation('sta2', mac='00:00:00:00:00:03', ip='10.0.0.3/8',
                          range=20)
    ap1 = net.addAccessPoint('ap1', ssid='ssid-ap1', mode='g', channel='1',
                             position='15,30,0', range=30)
    ap2 = net.addAccessPoint('ap2', ssid='ssid-ap2', mode='g', channel='6',
                             position='55,30,0', range=30)
    c1 = net.addController('c1')

    net.setPropagationModel(model="logDistance", exp=5)

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

    info("*** Creating links\n")
    net.addLink(ap1, ap2)

    net.plotGraph(max_x=100, max_y=100)

    net.startMobility(time=0)
    net.mobility(sta1, 'start', time=1, position='10,30,0')
    net.mobility(sta2, 'start', time=2, position='10,40,0')
    net.mobility(sta1, 'stop', time=10, position='60,30,0')
    net.mobility(sta2, 'stop', time=10, position='25,40,0')
    net.stopMobility(time=11)

    info("*** Starting network\n")
    net.build()
    c1.start()
    ap1.start([c1])
    ap2.start([c1])

    info("*** Running CLI\n")
    CLI(net)

    info("*** Stopping network\n")
```

```
    net.stop()


if __name__ == '__main__':
    setLogLevel('info')
    topology()
```

Execute o script. Em seguida, no terminal do Mininet-WiFi digite:

```
sta1 ping sta2
```

> Caso o handover esteja rápido demais, altere os os valores de "time" para estação sta1.

> Atenção: Caso o nome do arquivo esteja com extensão .txt, renomeie o arquivo para tirar o .txt.

> P: A comunicação entre sta1 e sta2 se manteve ininterrupta? Por quê? O que aconteceu com sta1?
>
> P: Qual o nome do processo que a mobilidade ocasionou em sta1 ? Comente como ele funciona.

Atividade (Extra): Wireless Mesh x Adhoc

Parte 1

Até o momento, somente redes infra-estruturadas foram exploradas. Nesse tipo de rede a transferência de dados acontece sempre entre uma estação e um ponto de acesso, ou AP (Access Point). Os APs são nós responsáveis pela captura e retransmissão das mensagens enviadas pelas estações. A transferência de dados nunca ocorre diretamente entre duas estações.

Agora, vamos explorar 2 outros tipos de redes sem fio, as redes sem fio mesh e adhoc, onde não há a presença do nó central, o AP. Para responder as perguntas a seguir considere o script adhoc_mesh.py disponível no diretório /lab3.

> P: O que ocorre quando sta1 tenta se comunicar com sta3? E sta4 com sta6? Comente e justifique o comportamento.
>
> P: Crie rotas estáticas de forma que sta4 consiga se comunicar com sta6. Descreva abaixo os comandos utilizados para a criação das rotas.

> Dica: Comandos comuns ao sistema operacional linux podem ser utilizados, como o ip route). Considere utilizá-los no terminal do Mininet-WiFi. Assim você terá garantias que seus comandos funcionarão como esperado através de testes que você mesmo poderá realizar após aplicar as regras de roteamento. Observe que a configuração default dos sistemas finais não tem encaminhamento de datagramas IP, pesquise como alterar a configuração do net.ipv4 para tornar roteadores finais em roteador.

Parte 2

Agora vamos realizar uma mudança simples no código de forma que você possa gerar um cenário 3D. Para tanto, altere os seguintes trechos no código:

De: `sta1 = net.addStation('sta1', position='10,10,0')`
Para: `sta1 = net.addStation('sta1', position='10,10,200')`

De: `net.plotGraph(max_x=200, max_y=200)`
Para: `net.plotGraph(max_x=200, max_y=200, max_z=200)`

> P: Como pode ver sta1 agora parece estar incomunicável, pois está fora do raio de alcance de sta2 e sta3. Para quais tipos de experimentos você utilizaria a visualização 3D?

Apêndices

Fundamentação teórica: Qual a diferença entre o Mininet e o Mininet-WiFi?

A emulação de redes tem sido bastante utilizada na avaliação de desempenho, em testes depuração de protocolos e também em diversos assuntos relacionados a pesquisas em redes de computadores. Dentro desse universo

encontra-se a emulação de redes sem fio.

Devido a diversos fatores, como modulação, interferência eletromagnética, mobilidade, a emulação de redes sem fio não é uma tarefa fácil, mas quando implementada em um ambiente controlado pode possibilitar projetos e desenvolvimentos de novos protocolos e aplicações para redes WiFi com alta fidelidade.

Visando proporcionar esse ambiente controlado e capaz de prover alta fidelidade, este laboratório baseia-se no Mininet-WiFi, uma extensão do emulador Mininet com suporte ao WiFi. Com ele é possível a virtualização de estações e pontos de acesso, e também utilizados nós já presentes do Mininet, como computadores, comutadores e controladores Open-Flow.

Arquitetura

Todo o processo de virtualização do Mininet-WiFi funciona de forma similar ao Mininet,tendo como base processos que são executados em network namespaces e placas de rede virtuais. Em nossa implementação, as interfaces sem fio virtualizam o hardware WiFi, que a depender do modo de funcionamento (estação ou ponto de acesso), pode funcionar nos modos managed e master, respectivamente.

As estações se comunicam com os pontos de acesso através de um processo chamado de autenticação e associação. Em nossa implementação cada estação possui uma interface sem fio (por padrão, podendo serem adicionadas mais). Uma vez associada a um ponto de acesso, as estações podem se comunicar com os tradicionais hosts do Mininet (tratamos como computadores na escrita deste trabalho), caso estes estiverem também conectados ao ponto de acesso (que é um switch virtual com capacidades WiFi) através de um meio cabeado. Já os pontos de acesso são responsáveis pela gestão das estações que estão associadas a ele.

Os pontos de acesso são virtualizados através do daemon hostapd, [1] que basicamente utilizam interfaces WiFi virtuais para prover capacidades de um ponto de acesso. Detalhes sobre o ambiente de execução do Mininet-WiFi são tratadas na subseção a seguir.

Fluxo de trabalho

O processo de funcionamento do Mininet-WiFi ocorre da seguinte forma: durante a inicialização de uma topologia, primeiramente ele verifica se a topologia é uma topologia WiFi.Se (http://WiFi.Se) não for, ele irá funcionar exatamente com os mesmos recursos que o Mininet oferece. Por outro lado, se for uma topologia WiFi, o Mininet-WiFi deverá carregar um módulo chamado mac80211_hwsim com o número de interfaces sem fio virtuais definidas pelo usuário (padrão é 3). Com o módulo carregado, o daemon hostapd também é executado para prover os serviços de ponto de acesso. Antes de se comunicarem com o SoftMac, as estações e pontos de acesso se comunicam com uma API chamada de cfg80211, que por sua vez se comunica com uma framework chamada mac80211. É esta framework que interage com o SoftMac através de um sock, o nl80211.

A maioria dos dispositivos wireless para Linux utilizam o SoftMac e suportam a maioria dos recursos oferecidos pelas interfaces wireless. Para iniciar uma topologia com duas estações e um ponto de acesso, basta utilizar o comando `mn –wifi –ssid=new-ssid` na CLI do Mininet-WiFi. Como esse comando as duas estações são automaticamente associadas ao ponto de acesso através do SSID new ssid e um controlador OpenFlow também é iniciado e responsável pelo controle de fluxos do ponto de acesso. Além da utilização de comandos, também é possível construir topologias através de alguns arquivos de exemplo que estão dentro da pasta examples no código do Mininet-WiFi. Acreditamos que os exemplos são importantes e serão facilitadores para os usuários que desejam criar novas topologias. Além da CLI, também é possível criar topologias para o Mininet-WiFi utilizando o Visual Network Description, disponível em http://ramonfontes.com/vnd (http://ramonfontes.com/vnd).

Interagindo com o ambiente de emulação

O Mininet-WiFi mantêm a mesma estrutura de interação do Mininet, por exemplo: para realizar testes de conectividade entre dois nós de rede, comandos como `sta1 ping sta2` ou `sta1 iperf -s & sta2 iperf -c 10.0.0.1` podem ser utilizados, respectivamente para testes de conectividade ou para mensurar largura de banda. Adicionalmente, outros comandos podem ser utilizados para melhor experiência com o ambiente WiFi, como: `sta1 iw dev sta1-wlan0 scan` para que uma estação sta1 busque as redes que estão ao seu alcance ou `sta1 iw dev sta1-wlan0 connect my-ssid` para possibilitar a

associação de uma estação sta1 a um ponto de acesso com SSID chamado my-ssid. Todos os novos comandos disponíveis no Mininet-WiFi são provenientes do iw que substitui o já conhecido iwconfig em sistemas Linux.

Mais informações

• Repositório do Livro Emulando Redes sem Fio com Mininet-WiFi: https://github.com/ramonfontes/mn-wifi-book-pt (https://github.com/ramonfontes/mn-wifi-book-pt)
• Repositorio Github e Wiki: https://github.com/intrig-unicamp/mininet-wifi (https://github.com/intrig-unicamp/mininet-wifi)
• Manual do Usuário: disponível na página do código fonte.
• Demonstrações e Videos: https://github.com/intrig-unicamp/mininet-wifi/wiki/Demos (https://github.com/intrig-unicamp/mininet-wifi/wiki/Demos)

Wireshark e a captura de pacotes 802.11

802.11 (Wireless LAN) e 802.3 (Ethernet LAN) são dois protocolos da família de padrões IEEE 802 LAN/MAN que usam acesso a um canal de comunicação compartilhado e endereçamento de camada de enlace (Medium Access Control - MAC) com endereços do tipo Ethernet formados por 6 octetos (48-bits). Apesar da semelhança, devido à complexidade adicional das redes 802.11, os frames desta rede e a frequência com a qual são trocados são maiores quando comparados a uma típica rede 802.3.

Por essas e outras, a captura de pacotes em redes Wireless utilizando o Wireshark e TShark resume-se aos pacotes de dados (exclui-se pacotes de controle e gerenciamento, por exemplo). Mais do que isso, antes de serem enviados ao host ou SO, os pacotes são destituídos de sua natureza Wireless (retira-se o cabeçalho 802.11) e colocados em um cabeçalho Ethernet "falso" pelo driver/hardware de rede do sistema. Então, podemos dizer que ocorre uma filtragem e tradução dos pacotes realmente passando pela interface. Tudo isso para facilitar nossa vida na hora de observar o tráfego da rede. Lembrando que esse comportamento varia dependendo do seu adaptador de rede e SO (Sistema Operacional).

Caso quisermos observar os pacotes Wireless em sua completude, teremos que habilitar o monitor mode na interface Wireless. Melhor ainda, para não interferir com nossa interface atual, vamos criar uma interface auxiliar que possua o

monitor mode habilitado. O monitor mode desabilita a filtragem e tradução dos pacotes 802.11, o que possibilita a visualização de todos os pacotes de todas as SSIDs do canal de captura selecionado.

Chega de papo. Para criar uma interface monitor auxiliar no dispositivo desejado utilizamos os comandos:

```
iw dev <interface de captura> interface add <interface monitor> type monitor
```

```
ifconfig <interface monitor> up
```

Por exemplo, podemos fazer isso diretamente da CLI do Mininet-WiFi para o um Ponto de Acesso (ap1) utilizando:

```
ap1 iw dev ap1-wlan1 interface add ap1-mon type monitor
```

```
ap1 ifconfig ap1-mon up
```

Ou para uma Estação (sta1) com:

```
sta1 iw dev sta1-wlan0 interface add sta1-mon type monitor
```

```
sta1 ifconfig sta1-mon up
```

Pronto! Agora é só abrir o Wireshark e iniciar a captura na interface criada (e.g., ap1-mon, sta1-mon).

Mais Informações

https://wiki.wireshark.org/CaptureSetup/WLA
(https://wiki.wireshark.org/CaptureSetup/WLA)

1. Hostapd (HostAccessPointDaemon) é um software a nível de usuário capaz de tonar uma interface de rede sem fio em pontos de acesso e servidores de autenticação. ↩