Understanding Networks: Mininet, Adhoc, and WiFi

Let's explore three key terms in networking: Mininet, Adhoc networks, and WiFi networking.

* **Mininet:** Imagine a network simulator that creates virtual networks on your computer. That's Mininet! It's a popular tool for researchers and network engineers to design, test, and experiment with new network protocols and architectures. Mininet helps build virtual networks with software instead of relying on physical hardware, making it fast, flexible, and cost-effective.
* **Adhoc Networks:** Picture a group of devices connecting directly with each other, without needing a central access point like a router. This is an Adhoc network. Think of laptops setting up a temporary network to share files or play games. In Adhoc mode, devices communicate peer-to-peer, relying on each other to route messages if needed.
* **WiFi Networking:** This is the familiar wireless networking technology that connects devices like laptops, phones, and tablets to the internet using Wi-Fi routers. WiFi uses radio waves to transmit data, allowing devices to connect without cables. WiFi networks can be secured with passwords to control access.

Here's a quick comparison:

| **Feature** | **Mininet** | **Adhoc Network** | **WiFi Network** |
| --- | --- | --- | --- |
| Purpose | Simulate real networks | Direct device-to-device connection | Wireless internet access |
| Infrastructure | Software-based | No central access point | Requires a router |
| Use Case | Testing network protocols | Temporary file sharing | Everyday internet access |

In short, Mininet provides a virtual playground to experiment with networks, Adhoc allows direct device communication, and WiFi offers wireless internet connectivity.

**1. Imports (Lines 1-3):**

Python

from mininet.node import Host

from mininet.wifi import Station, Link

from mininet.cli import CLI

This section imports the necessary libraries for building and interacting with the Mininet-WiFi network:

* mininet.node.Host: Used for creating wired hosts (not used in this script for ad hoc networks).
* mininet.wifi.Station: Used for creating wireless stations (devices) in the ad hoc network.
* mininet.wifi.Link: Used for creating physical connections (optional) between stations.
* mininet.cli.CLI: Provides access to the command-line interface within the script.

**2. Network Parameters (Lines 5-8):**

# Define network parameters (modify these based on your assignment)

num\_stations = 3 # Number of stations in the ad hoc network

ssid = 'adhoc\_network' # Service Set Identifier

channel = 1 # Wireless channel

This section defines variables that configure the overall wireless network:

* num\_stations: Sets the number of stations to be created in the ad hoc network. Modify this based on your assignment.
* ssid: Defines the Service Set Identifier (network name) for the ad hoc network.
* channel: Specifies the wireless channel on which the stations will communicate.

**3. Device Configurations (Lines 10-13):**

# Define device configurations (replace with data from your table)

sta\_macs = ['00:00:00:00:00:01', '00:00:00:00:00:02', '00:00:00:00:00:03']

sta\_ips = ['10.0.0.1', '10.0.0.2', '10.0.0.3']

This section defines lists containing configuration details for each station:

* sta\_macs: A list containing MAC addresses for each station. Replace these with actual MAC addresses from your assignment table.
* sta\_ips: A list containing IP addresses for each station. Replace these with actual IP addresses from your table.

**4. Create Network (Line 15):**

# Create Mininet network

net = mininet.net.Mininet(waitConn=True)

This line initializes a Mininet network object. The waitConn=True argument ensures all stations are connected before proceeding.

**5. Create Stations (Lines 17-24):**

# Create stations (modify for additional configurations)

stations = []

for i in range(num\_stations):

sta = Station(f'sta{i+1}', mac=sta\_macs[i], ip=sta\_ips[i], ssid=ssid, channel=channel)

stations.append(sta)

net.addNode(sta)

This section creates the wireless stations in the ad hoc network:

* A loop iterates through the number of stations defined earlier.
* Inside the loop, a Station object is created for each station using the Station class from mininet.wifi.
  + The station object is named f'sta{i+1}' (e.g., sta1, sta2, etc.).
  + The MAC address, IP address, SSID, and channel are set based on the defined variables and data from your lists.
* Each created station object is appended to the stations list.
* The net.addNode(sta) call adds the station to the Mininet network.

**Advantageous Use of Adhoc Networks**

Adhoc networks can be beneficial in situations where a central infrastructure like a Wi-Fi router is unavailable or impractical to set up. Here are some examples:

* **Emergency Response:** First responders at a disaster site can create an adhoc network to share critical information and coordinate rescue efforts without relying on damaged communication infrastructure.
* **Remote Collaboration:** Researchers in a remote field location can establish an adhoc network to share data and communicate findings in real-time.
* **Temporary Events:** During conferences or events held outdoors, attendees can create an adhoc network to share files or internet access if a centralized Wi-Fi network is overloaded or unavailable.

**Real-World Examples:**

* In 2010, following the devastating earthquake in Haiti, relief organizations utilized adhoc networks to establish communication channels in areas with damaged infrastructure. (<https://www.researchgate.net/publication/349926190_Mobile_Ad_Hoc_Network_in_Disaster_Area_Network_Scenario_A_Review_on_Routing_Protocols>)
* Mesh networks, a type of adhoc network technology, are being deployed in underserved communities to provide internet access in areas lacking traditional broadband infrastructure. (<https://en.wikipedia.org/wiki/Wireless_mesh_network>)

While adhoc networks offer flexibility and quick deployment, they might have limitations on security, scalability, and overall performance compared to traditional Wi-Fi networks.

* **OLSR (Optimized Link State Routing):** OLSR focuses on maintaining a global network view by exchanging routing information with neighboring nodes. In highly mobile adhoc networks, this overhead can lead to outdated routing tables and potentially impact TCP connections.
* **BATMAN (Better Approach to Mobile Ad-hoc Networking):** BATMAN utilizes a proactive routing approach with minimal routing information exchange. This can be more efficient in dynamic adhoc networks, potentially leading to higher TCP success rates compared to OLSR.