

# CSCI 572: Computer Networks II (Fall 2023)

## Mini-Project 3: Analyzing WiFi Behavior

In the class, we discussed WiFi's MAC layer protocol. The goal for this assignment is to study the behavior of WiFi infrastructure mode using ns-3. This project is a bit more open-ended than the previous two. One of the important skills to gain as a graduate student is how to approach open-ended research.

You need to do the following using Friis Propagation Loss Model.

**1. Create a network topology that will have the [hidden terminal issue](#); record nodes' RTS/CTS/ACK and data transmission times; analyze the recorded results; explain how WiFi's MAC protocol addresses the hidden terminal issue.**

**1.1 My simulation setup including parameters used, signal propagation model used, MAC layer protocol used, and network topology, etc in [hidden terminal](#)**

### (1) Simulation Parameters:

- **Number of Nodes (nWifis):** The simulation sets up 3 nodes.
- **Transmission Power (txp):** Nodes transmit with a power of 7.5 dBm.
- **Data Rate:** The OnOff application uses a data rate of 2048 bps.
- **Packet Size:** Packets sent from the OnOff application have a size of 64 bytes.
- **PHY Mode:** The PHY mode is set to "DsssRate11Mbps", which suggests the use of 802.11b DSSS with an 11 Mbps rate.
- **Simulation Duration (TotalTime):** The simulation runs for 100 seconds.
- **RTS/CTS Threshold:** Set to "0", meaning the RTS/CTS mechanism is enabled.

### (2) Signal Propagation Model:

- **Propagation Delay Model:** The simulation uses the ConstantSpeedPropagationDelayModel, which assumes that signals travel at the speed of light.
- **Propagation Loss Model:** The FriisPropagationLossModel is employed, which is a free-space model suitable for long distances, especially outdoors.

### (3) MAC Layer Protocol:

- **MAC Protocol Set Up:** ad hoc mode with AdhocWifiMac.
- **Rate Control:** disabled, and the simulation uses a constant data and control rate, as determined by the phyMode ("DsssRate11Mbps").
- **The RTS/CTS mechanism :** enabled, which is used to counteract the hidden terminal problem.

### (4) Network Topology and Mobility:

- **Node Positions:**

Node 0 is positioned at (0.0, 0.0, 0.0).

Node 1 is positioned at (200.0, 0.0, 0.0).

Node 2 is positioned at (100.0, 0.0, 0.0).

- **Mobility Model:** All nodes are set to use the ConstantPositionMobilityModel, meaning they do not move during the simulation. Their positions remain fixed at the specified coordinates.

The setup suggests that Node 2 is in between Node 0 and Node 1, which might be intended to create a scenario where the two outer nodes (Node 0 and Node 1) potentially act as hidden terminals to each other when trying to communicate with Node 2.

### (5) Routing Protocol:

The simulation uses the Ad hoc On-Demand Distance Vector (AODV) routing protocol for routing packets within the ad hoc network. AODV is a reactive routing protocol designed specifically for mobile ad hoc networks.

### (6) Applications:

Node 0 and Node 1 both run an OnOff application that sends UDP packets to Node 2. The OnOff application mimics traffic that switches between 'on' and 'off' periods, but given the OnTime and OffTime parameters, the source is constantly 'on'.

### (7) Output & Tracing:

The simulation outputs a CSV file (manet-routing.output.csv) with throughput metrics recorded every second. Additionally, PHY layer traces are recorded in "hide-wifi-phy.tr". A visual animation of the simulation is created using the NetAnim animator and is saved in "hide.xml".

In summary, My simulation sets up a basic three-node ad hoc network using 802.11b with the AODV routing protocol. The RTS/CTS mechanism is enabled, suggesting an interest in studying scenarios where the hidden terminal problem might arise. The setup, with two nodes transmitting to a central node, provides a topology conducive to such studies.

## 1.2 Record nodes' RTS/CTS/ACK and data transmission times in hidden terminal

### Extract RTS/CTS/ACK and data transmission time in hidden terminal

The following script processes the trace file "hide-wifi-phy.tr", extracts all lines containing the string "CTL\_RTS", saves those lines in a file named RTS\_lines.txt, and then prints a confirmation message to the terminal. This script is useful for quickly isolating specific trace events (in this case, RTS control frames) from a larger trace file.

```
1  #!/bin/bash
2
3  # input file name
4  input_file="hide-wifi-phy.tr"
5
6  # extract CTL, then put the result in .txt
7  grep "CTL_RTS" "$input_file" > RTS_lines.txt
8
9  # print the extract finished
10 echo "include the 'CTL_RTS' lines output to RTS_lines.txt"
```

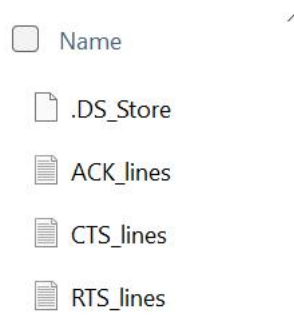
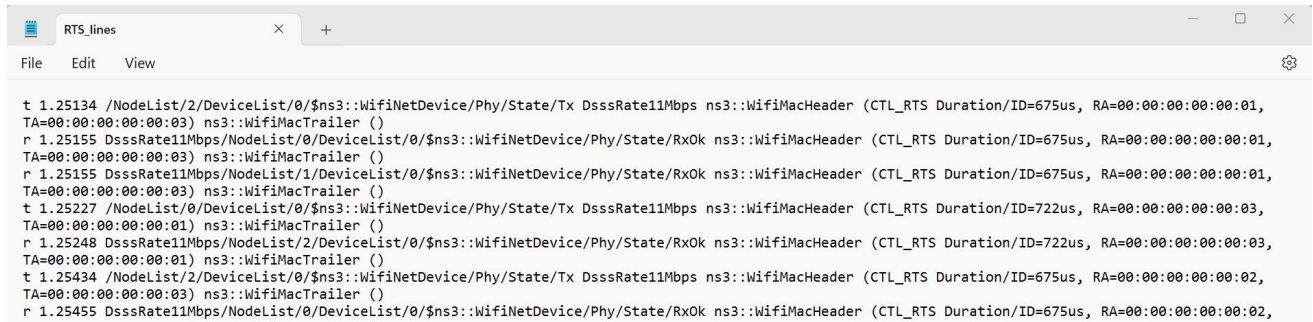


Figure1.1 Script for extract RTS/CTS/ACK in hidden terminal



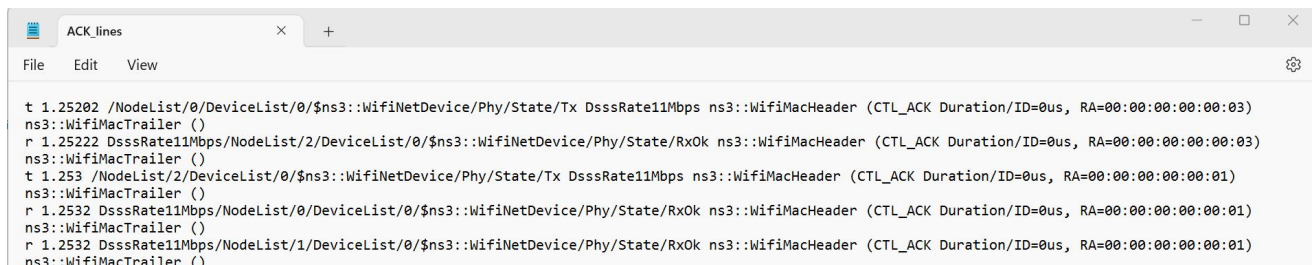
```
t 1.25134 /NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate11Mbps ns3::WifiMacHeader (CTL_RTS Duration/ID=675us, RA=00:00:00:00:00:01, TA=00:00:00:00:00:03) ns3::WifiMacTrailer ()
r 1.25155 DsssRate11Mbps/NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_RTS Duration/ID=675us, RA=00:00:00:00:00:01, TA=00:00:00:00:00:03) ns3::WifiMacTrailer ()
r 1.25155 DsssRate11Mbps/NodeList/1/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_RTS Duration/ID=675us, RA=00:00:00:00:00:01, TA=00:00:00:00:00:03) ns3::WifiMacTrailer ()
t 1.25227 /NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate11Mbps ns3::WifiMacHeader (CTL_RTS Duration/ID=722us, RA=00:00:00:00:00:03, TA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
r 1.25248 DsssRate11Mbps/NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_RTS Duration/ID=722us, RA=00:00:00:00:00:03, TA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
t 1.25434 /NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate11Mbps ns3::WifiMacHeader (CTL_RTS Duration/ID=675us, RA=00:00:00:00:00:02, TA=00:00:00:00:00:03) ns3::WifiMacTrailer ()
r 1.25455 DsssRate11Mbps/NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_RTS Duration/ID=675us, RA=00:00:00:00:00:02, TA=00:00:00:00:00:03) ns3::WifiMacTrailer ()
```

**Figure1.2 Extract result for RTS in hidden terminal**



```
t 3.25453 /NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate11Mbps ns3::WifiMacHeader (CTL_CTS Duration/ID=509us, RA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
r 3.25473 DsssRate11Mbps/NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_CTS Duration/ID=509us, RA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
r 3.25473 DsssRate11Mbps/NodeList/1/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_CTS Duration/ID=509us, RA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
t 3.50213 /NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate11Mbps ns3::WifiMacHeader (CTL_CTS Duration/ID=509us, RA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
r 3.50233 DsssRate11Mbps/NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_CTS Duration/ID=509us, RA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
r 3.50233 DsssRate11Mbps/NodeList/1/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_CTS Duration/ID=509us, RA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
```

**Figure1.3 Extract result for CTS in hidden terminal**



```
t 1.25202 /NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate11Mbps ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:03) ns3::WifiMacTrailer ()
r 1.25222 DsssRate11Mbps/NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:03) ns3::WifiMacTrailer ()
t 1.253 /NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate11Mbps ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
r 1.2532 DsssRate11Mbps/NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
r 1.2532 DsssRate11Mbps/NodeList/1/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:01) ns3::WifiMacTrailer ()
```

**Figure1.4 Extract result for ACK in hidden terminal**

The provided screenshots displays contents of a file named RTS/CTS/ACK\_lines that be a trace output from an ns-3 network simulation, particularly related to the Clear To Send (CTS) mechanism in Wi-Fi communication. Both Node 0 and Node 1 are successfully receiving the CTS frames sent by Node 2.

This **Figure1.2 Extract result for RTS in hidden terminal** provides insights into the Wi-Fi communication mechanisms in use, particularly the RTS/CTS handshake that helps in mitigating the hidden node problem in wireless networks. The CTS frames are sent in response to a prior Request To Send (RTS) frame (not shown in this snippet) to notify the sender that it can begin data transmission.

**Here's a breakdown of the trace contents:**

#### Event Type:

- t: Denotes a transmission event.
- r: Denotes a reception event.
- Time: Each line starts with a time stamp (e.g., 3.25453). This represents the simulation time at which the event (transmission or reception) took place.

#### Node & Device Information:

- /NodeList/2/DeviceList/0/\$ns3::WifiNetDevice/Phy/State/Tx: This indicates the specific node (NodeList/2 = Node 2) and device (DeviceList/0 = first or only Wi-Fi device on the node) involved in the event. The event here is a transmission (Tx).
- /NodeList/0/DeviceList/0/\$ns3::WifiNetDevice/Phy/State/RxOk: This indicates Node 0's first or only Wi-Fi device successfully receiving (RxOk) a packet.

This trace provides insights into the Wi-Fi communication mechanisms in use, particularly the RTS/CTS handshake that helps in mitigating the hidden node problem in wireless networks. The CTS frames are sent in response to a prior Request To Send (RTS) frame (not shown in this snippet) to notify the sender that it can begin data transmission.

### 1.3 Visualize and Analyze the Recorded Results in hidden terminal

#### Nodes:

- There are three nodes labeled 0, 1, and 2.
- Node 0 is situated at position (0,0,0) on the grid.
- Node 1 and Node 2 are to the right on the grid.

#### Communication:

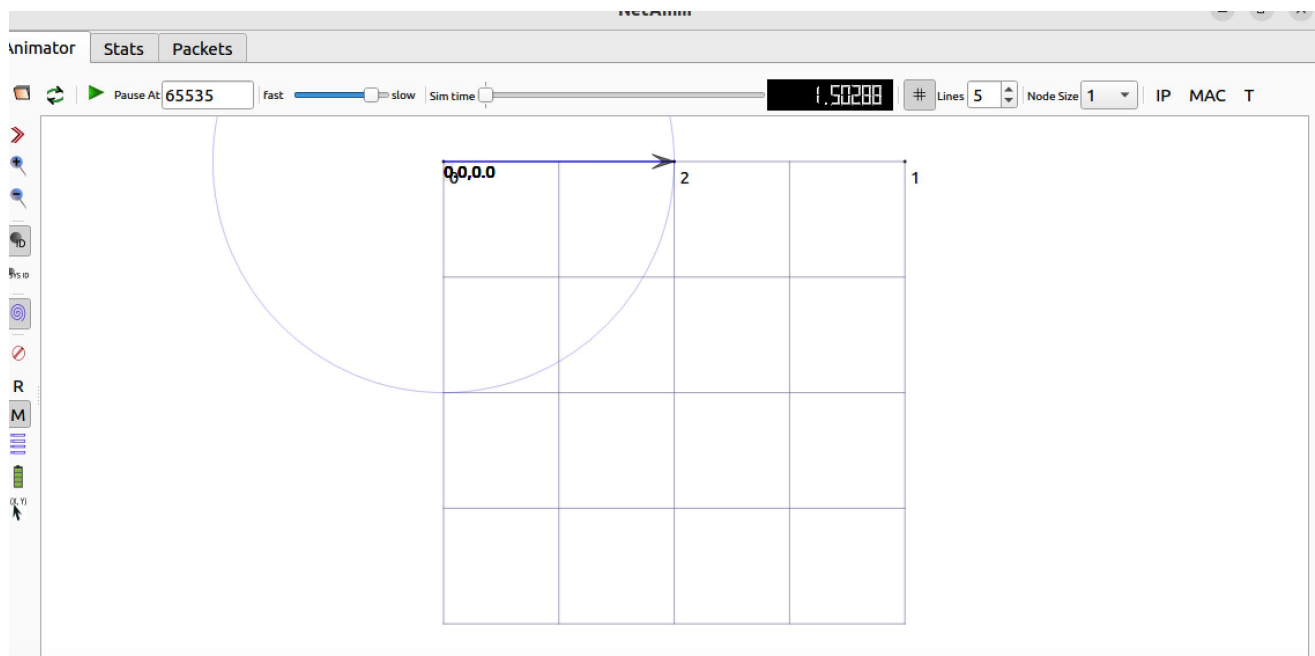
Two arrows are visible:

- A blue arrow starting from Node 0 and pointing towards Node 2, indicating communication or data being sent from Node 0 to Node 2.
- A black arrow starting from Node 2 and pointing towards Node 1, suggesting communication or data transfer from Node 2 to Node 1.

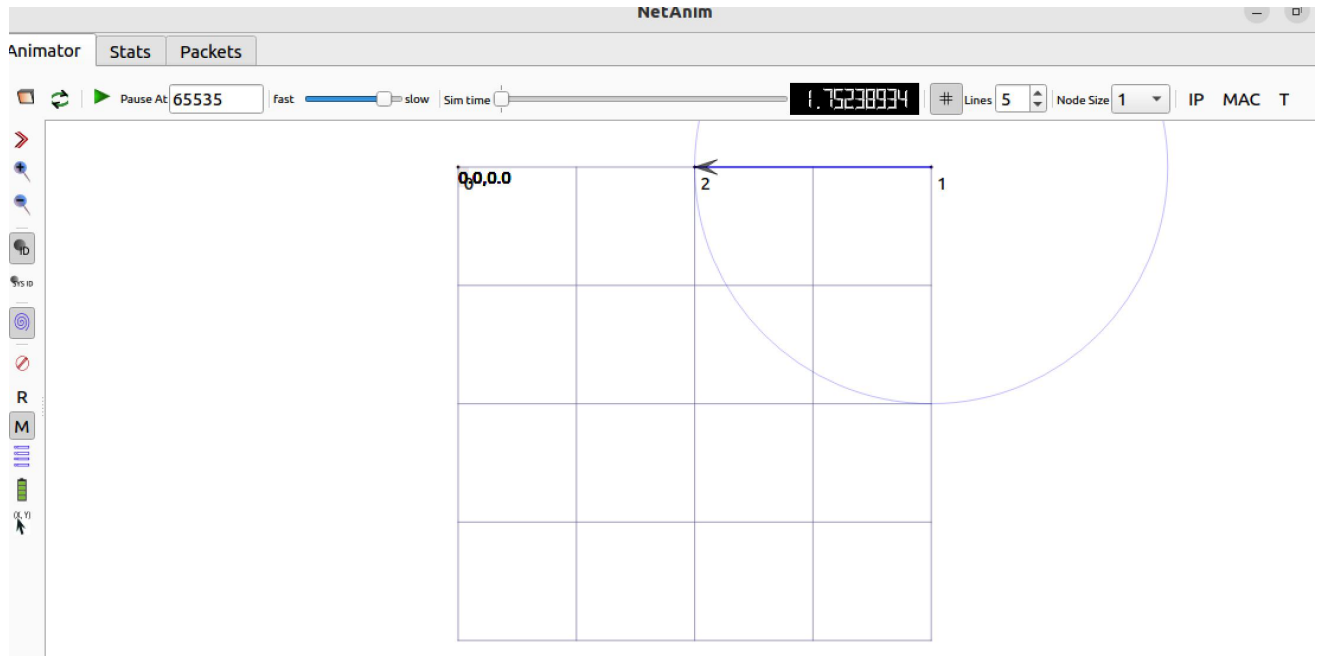
#### Animator Control:

The current simulation time appears to be paused at 65535, as reflected in the "Pause At" field.

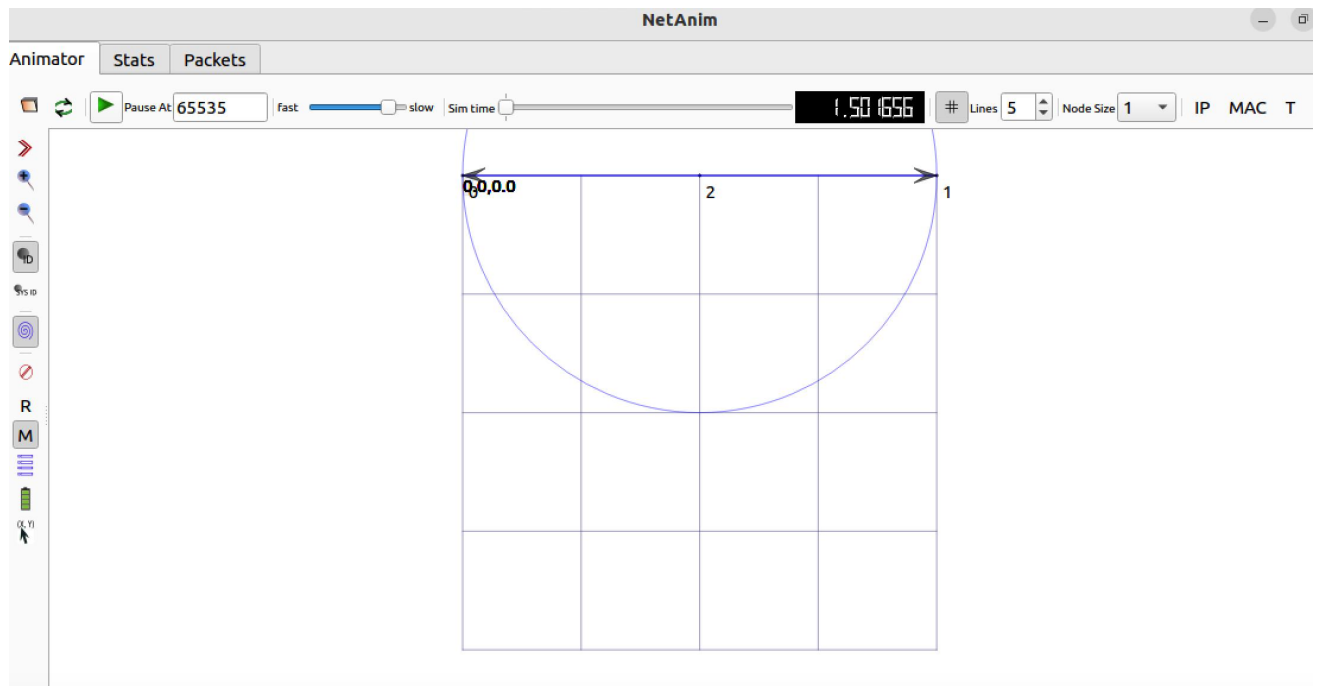
Playback controls such as play, pause, fast-forward, and rewind are present at the top, allowing you to navigate through the simulation timeline.



**Figure1.5 Visualization the Recorded Results-01 in hide terminal**



**Figure1.6 Visualization the Recorded Results-02 in hide terminal**



**Figure1.7 Visualization the Recorded Results-03 in hide terminal**

In summary, this visualizations displays a three-node network scenario. Node 0 is sending data to Node 2, and in turn, Node 2 is forwarding or sending data to Node 1. This could be a relay or multi-hop scenario, where data travels through an intermediate node (Node 2 in this case) before reaching its final destination (Node 1). The positions of the nodes and the arrows displaying communication provide insights into the communication patterns within this network simulation.

## 1.4 Explain how WiFi's MAC protocol addresses the hidden terminal issue

Here's how the RTS/CTS mechanism helps in resolving the hidden terminal issue:

### RTS (Request to Send):

Before sending a data packet, the transmitting node sends an RTS frame to the intended receiver. The RTS frame is short and contains information about the upcoming transmission, such as its duration.

### CTS (Clear to Send):

Upon receiving the RTS, the receiver responds with a CTS frame if it's ready to receive the data packet. The CTS frame also includes information about the duration of the upcoming transmission.

### Channel Reservation:

Both RTS and CTS frames act as alerts to neighboring nodes about the upcoming transmission. When other nodes receive these frames, they parse the duration information and set their Network Allocation Vectors (NAV) to defer their own transmissions for that specified duration. This effectively "reserves" the wireless medium for the intended transmission and reduces the chances of a collision.

### Actual Data Transmission:

Once the CTS is received by the sender, it knows that the path is clear, and the receiver is ready. The sender then transmits the actual data packet.

### Acknowledgment:

After successfully receiving the data packet, the receiver sends an acknowledgment (ACK) to inform the sender that the data was received correctly.

The RTS/CTS mechanism is beneficial, especially in environments with high traffic and where the hidden terminal problem is prevalent. However, there is some overhead involved. The exchange of RTS and CTS frames introduces additional delay and consumes bandwidth. Therefore, in some scenarios, especially when data packets are very short, the RTS/CTS mechanism might be turned off to optimize performance.

It's worth noting that while the RTS/CTS mechanism can alleviate the hidden terminal problem, it cannot completely eliminate all collisions in a wireless environment due to other challenges like the exposed terminal problem and external interferences.

## 2. Create a network topology that will have the exposed terminal issue; record nodes' RTS/CTS/ACK and data transmission times; analyze the recorded results; explain how WiFi's MAC protocol addresses the exposed terminal issue.

### 2.1 My simulation setup including parameters used, signal propagation model used, MAC layer protocol used, and network topology in exposed terminal issue

#### (1) Simulation Parameters:

- **Number of Nodes:** 4 nodes (nWifis = 4).
- **Simulation Time:** 100 seconds (TotalTime = 100.0).
- **Traffic Type:** On-Off traffic with a data rate of 2048bps (rate).
- **Packet Size:** 64 bytes (PacketSize).
- **WiFi Standard:** 802.11b.



- **PHY Mode:** DSSS Rate 11Mbps (phyMode).
- **Transmission Power:** 7.5 dBm (txp = 7.5).

## (2) Signal Propagation Model:

- **Propagation Delay Model:** Constant Speed (ns3::ConstantSpeedPropagationDelayModel).
- **Propagation Loss Model:** Friis (ns3::FriisPropagationLossModel).

## (3) MAC Layer Protocol:

- **Type:** Ad hoc WiFi MAC (ns3::AdhocWifiMac).
- **Rate Control:** Rate control is disabled, and a constant rate is set for both data and control packets (using phyMode).
- **RTS/CTS Mechanism:** Enabled (since RtsCtsThreshold is set to "0").

## (4) Network Topology:

### Node Positions:

- Node 0: Position (0.0, 0.0, 0.0).
- Node 1: Position (50.0, 0.0, 0.0).
- Node 2: Position (150.0, 0.0, 0.0).
- Node 3: Position (200.0, 0.0, 0.0).
- **Mobility Model:** All nodes use the ns3::ConstantPositionMobilityModel, meaning they remain stationary throughout the simulation.

## (5) Routing Protocol:

- **Type:** AODV (AodvHelper aodv).

## (6) Traffic Setup:

- Node 0 acts as a sink, receiving packets from Node 1.
- Node 3 acts as a sink, receiving packets from Node 2.
- On-Off applications are installed on nodes 1 and 2 to send packets to nodes 0 and 3, respectively.

## (7) Trace Files and Outputs:

- **ASCII Trace:** Outputs to a file named "expose-wifi-phy.tr".
- **Throughput Metrics:** Outputs to a CSV file (m\_CSVfileName).
- **Animation File:** Outputs to a file named "expose.xml" for visualization using NetAnim.

## 2.2 Record nodes' RTS/CTS/ACK and data transmission times in expose terminal

Extract RTS/CTS/ACK and data transmission time in expose terminal

The following script processes the trace file "expose-wifi-phy.tr", extracts all lines containing the string "CTL\_RTS", saves those lines in a file named RTS\_lines.txt, and then prints a confirmation message to the terminal. This script is useful for quickly isolating specific trace events (in this case, RTS control frames) from a larger trace file.

```

1  #!/bin/bash
2
3  # input file name
4  input_file="expose-wifi-phy.tr"
5
6  # extract CTL, then put the result in .txt
7  grep "CTL_RTS" "$input_file" > RTS_lines.txt
8
9  # print the extract finished
10 echo "include the 'CTL_RTS' lines output to RTS_lines.txt"

```

.DS\_Store

RTS\_lines

CTS\_lines

ACK\_lines

expose-wifi-phy.tr

**Figure2.1 Script for extract RTS/CTS/ACK in exposed terminal**

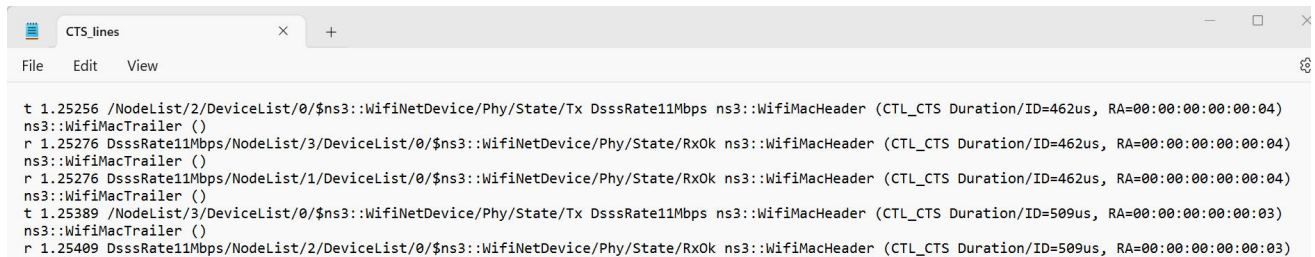


```

t 1.25234 /NodeList/3/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate1Mbps ns3::WifiMacHeader (CTL_RTS Duration/ID=675us, RA=00:00:00:00:00:03,
TA=00:00:00:00:00:04) ns3::WifiMacTrailer ()
r 1.25255 DsssRate1Mbps/NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_RTS Duration/ID=675us, RA=00:00:00:00:00:03,
TA=00:00:00:00:00:04) ns3::WifiMacTrailer ()
t 1.25367 /NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate1Mbps ns3::WifiMacHeader (CTL_RTS Duration/ID=722us, RA=00:00:00:00:00:04,
TA=00:00:00:00:00:03) ns3::WifiMacTrailer ()
r 1.25388 DsssRate1Mbps/NodeList/3/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_RTS Duration/ID=722us, RA=00:00:00:00:00:04,
TA=00:00:00:00:00:03) ns3::WifiMacTrailer ()
r 1.25388 DsssRate1Mbps/NodeList/1/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_RTS Duration/ID=722us, RA=00:00:00:00:00:04,
TA=00:00:00:00:00:03) ns3::WifiMacTrailer ()
t 1.25634 /NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate1Mbps ns3::WifiMacHeader (CTL_RTS Duration/ID=675us, RA=00:00:00:00:00:02,

```

**Figure2.2 Extract result for RTS in exposed terminal**

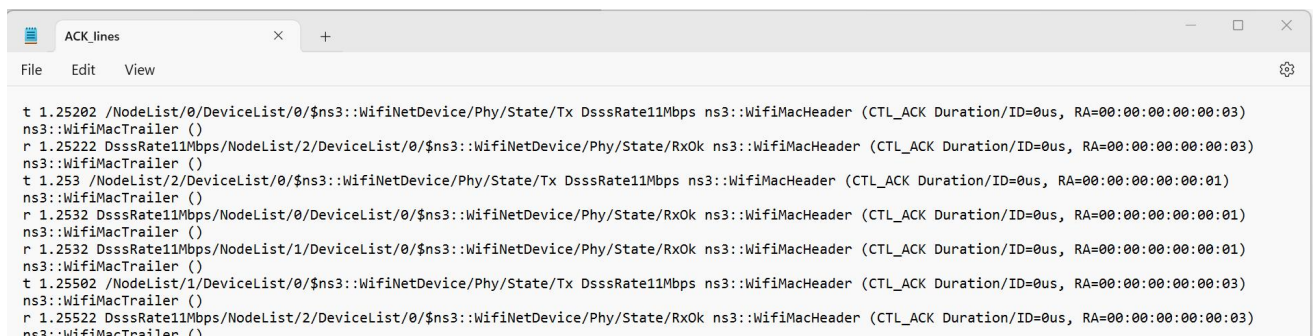


```

t 1.25256 /NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate1Mbps ns3::WifiMacHeader (CTL_CTS Duration/ID=462us, RA=00:00:00:00:00:04)
ns3::WifiMacTrailer ()
r 1.25276 DsssRate1Mbps/NodeList/3/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_CTS Duration/ID=462us, RA=00:00:00:00:00:04)
ns3::WifiMacTrailer ()
r 1.25276 DsssRate1Mbps/NodeList/1/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_CTS Duration/ID=462us, RA=00:00:00:00:00:04)
ns3::WifiMacTrailer ()
t 1.25389 /NodeList/3/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate1Mbps ns3::WifiMacHeader (CTL_CTS Duration/ID=509us, RA=00:00:00:00:00:03)
ns3::WifiMacTrailer ()
r 1.25409 DsssRate1Mbps/NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_CTS Duration/ID=509us, RA=00:00:00:00:00:03)

```

**Figure2.3 Extract result for CTS in exposed terminal**



```

t 1.25202 /NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate1Mbps ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:03)
ns3::WifiMacTrailer ()
r 1.25222 DsssRate1Mbps/NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:03)
ns3::WifiMacTrailer ()
t 1.253 /NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate1Mbps ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:01)
ns3::WifiMacTrailer ()
r 1.2532 DsssRate1Mbps/NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:01)
ns3::WifiMacTrailer ()
r 1.2532 DsssRate1Mbps/NodeList/1/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:01)
ns3::WifiMacTrailer ()
t 1.25502 /NodeList/1/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx DsssRate1Mbps ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:03)
ns3::WifiMacTrailer ()
r 1.25522 DsssRate1Mbps/NodeList/2/DeviceList/0/$ns3::WifiNetDevice/Phy/State/RxOk ns3::WifiMacHeader (CTL_ACK Duration/ID=0us, RA=00:00:00:00:00:03)
ns3::WifiMacTrailer ()

```

**Figure2.4 Extract result for ACK in exposed terminal**

By examining the RTS messages and their timestamps, I can discern the attempt of nodes to access the medium and reserve it for their use.



This **Figure2.2 Extract result for RTS in exposed terminal** that successful reception (r) of an RTS at a node suggests that there wasn't a collision for that RTS. If you don't see a corresponding CTS message shortly after an RTS, it might imply a collision or that the receiving node is busy.

The "Duration/ID" field's value in the RTS message can give you an idea of the time the sender intends to occupy the channel. It reserves this time to prevent other nodes from sending and causing a collision.

Observing the sender and receiver nodes in consecutive lines can help identify communication patterns and potential contention or collisions.

Let's delve into the details of the provided output in **Figure2.2 Extract result for RTS in exposed terminal** extract result for RTS in expose terminal :

#### **Timestamp (e.g., t 1.25234):**

This represents the simulation time when the event occurred.

The character preceding the time (e.g., t or r) indicates the type of event:

- **t**: Transmission event.
- **r**: Reception event.

#### **Node Information (e.g., /NodeList/3/DeviceList/0/\$ns3::WifiNetDevice/Phy/State/Tx):**

- **/NodeList/3**: Refers to the node in the simulation. Here, it's node number 3.
- **DeviceList/0**: Denotes the first device on that node, typically the wireless interface.
- **\$ns3::WifiNetDevice/Phy/State/Tx**: Specifies the state of the device. In this instance, Tx indicates the node's state is "transmitting."

#### **Physical Layer Info (e.g., DsssRate11Mbps):**

This specifies the modulation and data rate being used. In the provided example, it indicates the use of 11 Mbps using Direct-sequence spread spectrum (DSSS) modulation.

#### **WiFi MAC Header Info (e.g., ns3::WifiMacHeader (CTL\_RTS Duration/ID=76s5us, RA=00:00:00:00:00:03)):**

- **CTL\_RTS**: Confirms that this frame is an RTS frame.
- **Duration/ID=765us**: This is the "duration" field in the RTS frame. It indicates the total time (in microseconds) reserved for the subsequent CTS, DATA, and ACK frames.
- **RA=00:00:00:00:00:03**: The "RA" or Receiver Address field denotes the intended recipient of the RTS frame. Here, it's directed to node 3.

#### **ns3::WifiMacTrailer ():**

- This represents the trailer section of the WiFi MAC frame. It typically contains error-checking information.

For addressing the exposed terminal issue, one would also need to observe the CTS, DATA, and ACK messages in a trace. The RTS/CTS mechanism can mitigate this issue by allowing a node to reserve the channel, ensuring that nearby nodes are aware and refrain from transmitting for the duration specified.

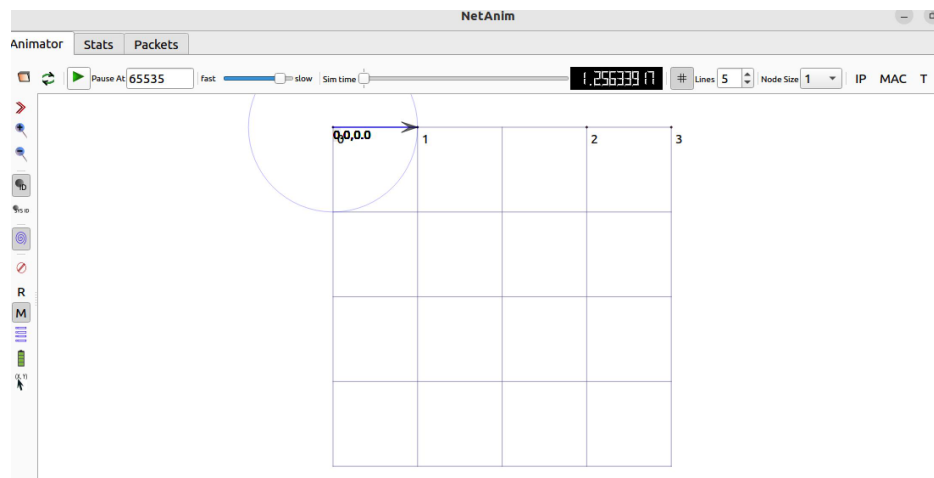
## 2.3 Visualize and Analyze the Recorded Results in expose terminal

This visualization helps in understanding the flow of packets, routing mechanisms, and potential issues in the exposed terminal problems.

**Nodes:** There are 4 nodes in the network, labeled 0, 1, 2, and 3.

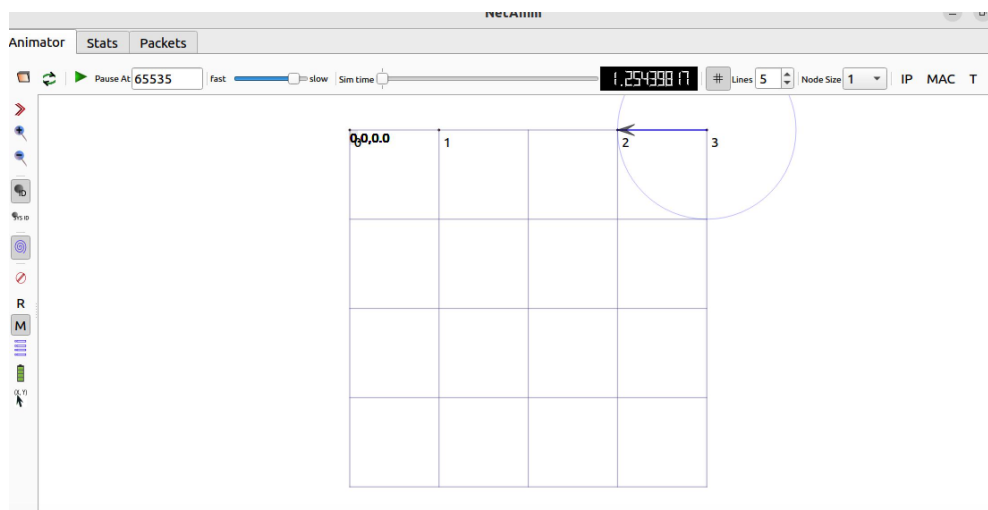
- Node 0 is on the far left.
- Node 1 is a bit to the right of Node 0.
- Node 2 is further to the right.
- Node 3 is on the far right.

The animation is visualizing a scenario in which nodes communicate in an ad-hoc manner using the AODV routing protocol, and the RTS/CTS mechanism is in use to exposed terminal problems.



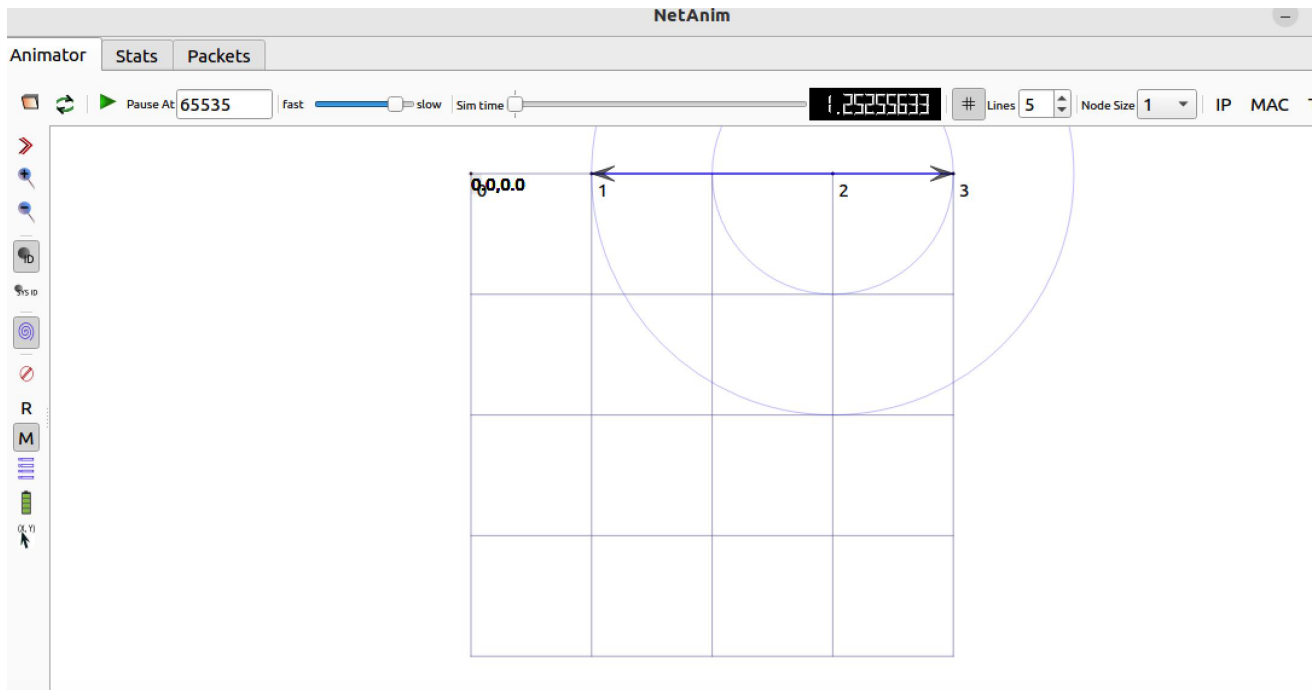
**Figure2.5 Visualization the Recorded Results-01 in exposed terminal**

In the **Figure2.5**, a packet is being sent from Node 1 towards Node 0.



**Figure2.6 Visualization the Recorded Results-02 in exposed terminal**

In the **Figure2.6** image, a packet is being sent from Node 2 to Node 3.



**Figure2.7 Visualization the Recorded Results-03 in exposed terminal**

In the **Figure2.7**, it appears that Node 1 is communicating with Node 2 and Node 2 is communicating with Node 3 simultaneously.

**Radii around Nodes:** The circles around each node likely represent the communication range or the transmission range of the nodes. If another node is within this circle, it can directly communicate with the node.

## 2.4 Explain how WiFi's MAC protocol addresses the **expose terminal issue**

The Exposed Terminal Problem occurs in wireless networks when a node incorrectly assumes that it cannot send packets to a certain destination because another nearby node is transmitting, even though the destination for both transmissions might be out of range of one another. As a result, the exposed node remains silent, leading to decreased network utilization.

The WiFi MAC protocol, specifically the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism, employs the RTS/CTS (Request to Send / Clear to Send) handshake to address this issue:

### **Request to Send (RTS):**

- Before a node sends data, it first sends an RTS packet.
- The RTS packet contains the intended receiver's address.

### **Clear to Send (CTS):**

- Upon receiving an RTS, the target node responds with a CTS packet.
- The CTS packet not only confirms that the receiving node is ready and clear to receive data but also indicates to other nodes in the vicinity to refrain from transmitting for a specified duration.

### How it Addresses the Exposed Terminal Problem:

- Consider nodes A, B, C, and D where A and C want to communicate with B and D respectively. B and D are out of each other's range, but A and C are within each other's range.
- When A wants to send data to B, it sends an RTS. B responds with a CTS.
- Node C hears the RTS from A and refrains from sending to D, thinking there will be a collision at B. However, when B sends the CTS, which C won't be able to hear since D is out of B's range, C realizes that it can safely send data to D without causing interference at B.

Thus, using the RTS/CTS handshake, both A and C can simultaneously send data to B and D, respectively, without any collisions, efficiently mitigating the exposed terminal problem.

### Additional Points:

- The RTS/CTS mechanism does introduce some overhead because of the handshake. Therefore, it's not always used in WiFi networks, especially if the data packets are small. The threshold to decide whether or not to use RTS/CTS is determined by the "RTS Threshold" parameter in the WiFi configuration.
- The RTS/CTS mechanism also addresses the "Hidden Terminal Problem" in wireless networks.

In summary, the RTS/CTS mechanism in the WiFi MAC protocol enables efficient utilization of the medium by resolving the exposed terminal problem, allowing nodes that were otherwise silent due to false collision assumptions to transmit data.

**3. Create a network topology where multiple nodes (at least three) are within proximity that will cause [wireless interference](#); design your experiments to study the impact of the number of interfering nodes affect the network throughput.**

**3.1 My simulation setup including parameters used, signal propagation model used, MAC layer protocol used, and network topology [in wireless interference](#)**

#### (1)Simulation Parameters:

- **TotalTime:** The total time for which the simulation runs is set to 100 seconds.
- **Rate:** The data rate for the OnOff application which sends the data is set to "2048bps".
- **phyMode:** Physical layer modulation rate is set to "DsssRate11Mbps" which corresponds to Direct Sequence Spread Spectrum rate of 11 Mbps, a typical setting for 802.11b.
- **txp:** The transmission power of the wifi nodes is 7.5 (not clear if this is dBm or some other unit as it's not explicitly mentioned).

#### (2)Signal Propagation Model:

- **The propagation delay:** ConstantSpeedPropagationDelayModel, which assumes a constant speed of light for signal propagation.

- **The propagation loss - 01:** [FriisPropagationLossModel](#), which is a commonly used model for free-space signal attenuation.
- **The propagation loss - 02:** [JakesPropagationLossModel](#), which represents a multipath fading model, typically used to describe fast fading due to the constructive and destructive interference of multiple signal paths.
- **(3)MAC Layer Protocol:**
- **The MAC type :**AdhocWifiMac. This means the nodes will communicate in a peer-to-peer manner without relying on an access point.
- **Rate control:** disabled by setting the data and control modes to a constant rate (phyMode).
- **RTS/CTS mechanism :** it's not active.

#### (4)Network Topology:

- **nWifis nodes** in my network, where nWifis will be provided as a command line argument. Node 0 is positioned at (0.0, 0.0, 0.0). **There are 5 nodes were set up by me ,the nodes numbered: 7, 9, 11, 13, 15.**
- **The other nodes:** placed in a grid starting from (10.0, 10.0, 0.0) with **a spacing of 5 units** in the X direction and 2 units in the Y direction. It have 2 nodes per row ("GridWidth", UIntegerValue (2)), but the total number of nodes are 14.
- **All nodes are set to have constant positions** (ns3::ConstantPositionMobilityModel).

#### (5)Routing Protocol:

- AODV (Ad hoc On-Demand Distance Vector) is used as the routing protocol for the nodes.

#### (6)Traffic:

- Node 0 is set to send data (via an OnOff application) to the last node (adhocNodes.Get (m\_wifis-1)).
- The data is sent over UDP.

#### (7)Output:

- **Throughput values** are written to a CSV file named "manet-routing.output.csv" by default, but the name can be modified through command-line arguments.
- **An animation trace file "ganrao.xml"** is created, which can be visualized using NetAnim.
- **The flow monitor** is used to gather statistics and calculate the average throughput, which is printed to the console.

In summary, this simulation is designed to study the throughput of a MANET of nWifis nodes arranged in a specific topology, using the AODV routing protocol and 802.11b WiFi standard.



## 3.2 Record the throughput values in wireless interference within [FriisPropagationLossModel](#) and [JakesPropagationLossModel](#)

A series of command line outputs related to the NS-3 simulation build and execution process.

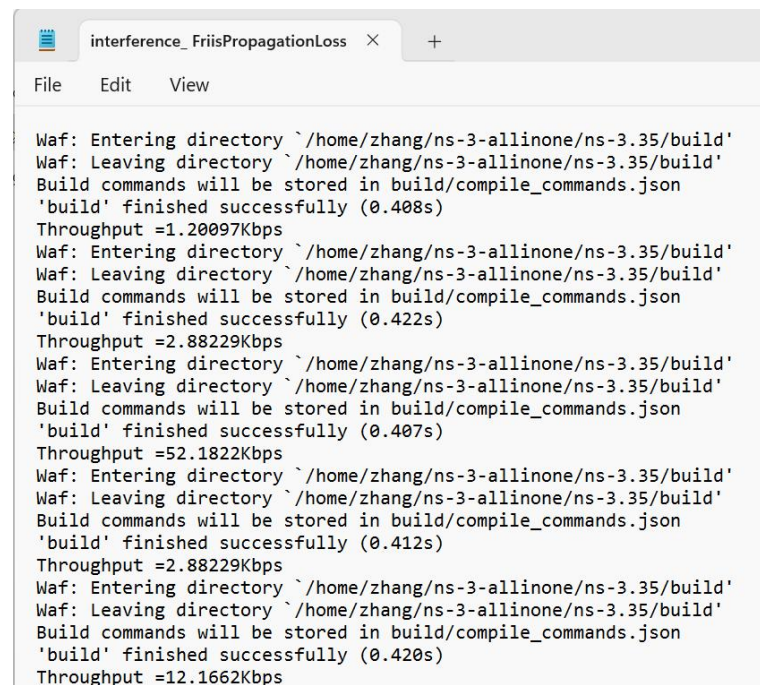
Here's a breakdown:

### (1) Building and Running:

- The output shows the NS-3 waf build system entering and leaving the directory `/home/zhang/ns-3-allinone/ns-3.35/build`. This is indicative of the compilation process for the simulation code.
- Each build process is followed by a statement about storing build commands in `build/compile_commands.json`.
- Each compilation appears to complete successfully within a time span of around 0.4 seconds (as indicated by times like (0.408s)).

### (2) Throughput:

- After each build and run of the simulation, the throughput is printed out.



```
Waf: Entering directory `/home/zhang/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/zhang/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.408s)
Throughput =1.20097Kbps
Waf: Entering directory `/home/zhang/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/zhang/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.422s)
Throughput =2.88229Kbps
Waf: Entering directory `/home/zhang/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/zhang/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.407s)
Throughput =52.1822Kbps
Waf: Entering directory `/home/zhang/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/zhang/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.412s)
Throughput =2.88229Kbps
Waf: Entering directory `/home/zhang/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/zhang/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.420s)
Throughput =12.1662Kbps
```

**Figure 3.1 Record throughput values in wireless interference using [FriisPropagationLossModel](#)**

These values are results of the simulation [using FriisPropagationLossModel](#), with varied throughput values reported: 1.20097Kbps, 2.88229Kbps, 52.1822Kbps, another 2.88229Kbps, and 12.1662Kbps. The variation in throughput might stem from changes in simulation parameters, evolving network conditions, or other dynamic factors in the simulation environment.

```

interference_JakesPropagationLos x +
File Edit View

Waf: Entering directory `/home/huang/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/huang/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.422s)
Throughput =2.88229Kbps
Waf: Entering directory `/home/huang/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/huang/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.421s)
Throughput =35.4096Kbps
Waf: Entering directory `/home/huang/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/huang/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.392s)
Throughput =2.88229Kbps
Waf: Entering directory `/home/huang/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/huang/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.429s)
Throughput =2.88229Kbps
Waf: Entering directory `/home/huang/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/huang/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.413s)
Throughput =23.7323Kbps

```

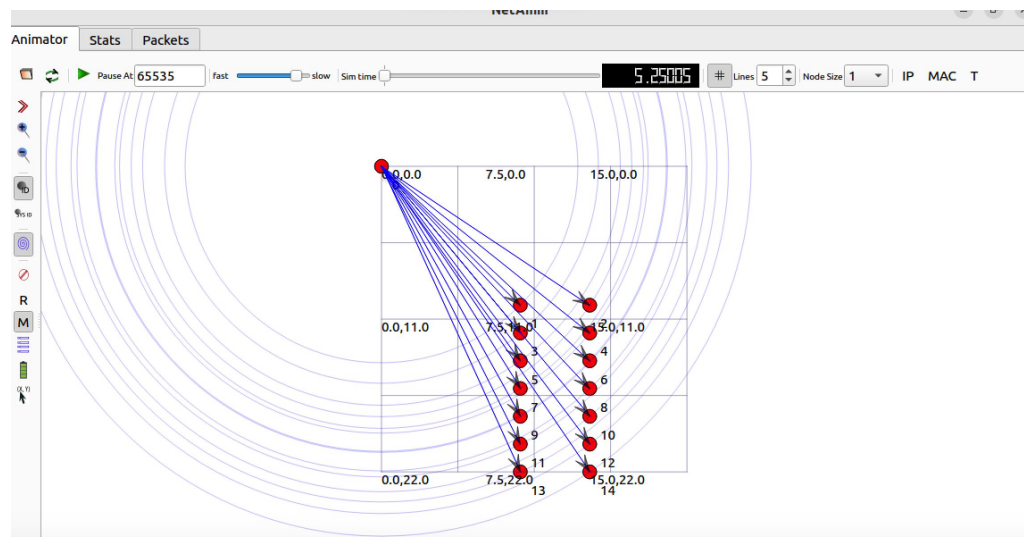
**Figure 3.2 Record throughput values in wireless interference using JakesPropagationLossModel**

Following each build and simulation using JakesPropagationLossModel , the throughput is displayed. We have the following throughput values for each run:2.88229Kbps, 35.4096Kbps, 2.88229Kbps, 2.88229Kbps, 23.7323Kbps.

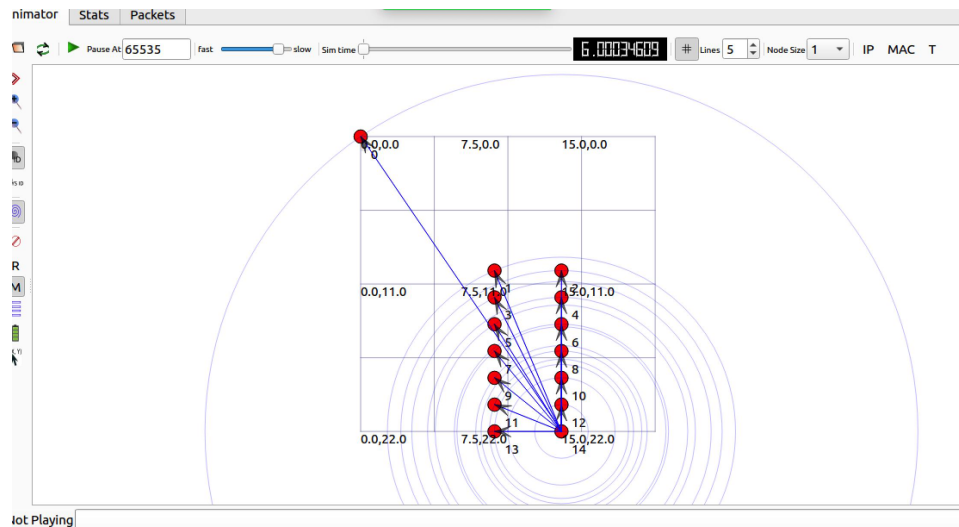
The variation in throughput might stem from changes in simulation parameters, evolving network conditions, or other dynamic factors in the simulation environment.

### 3.3 Visualize and Analyze throughput values in the wireless interference

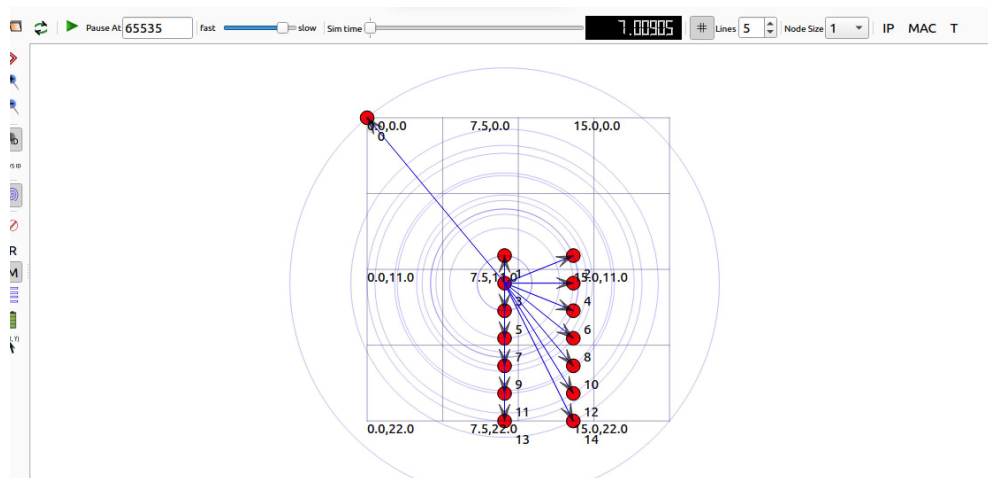
The simulation's visualization provides an intuitive understanding of the network's operation.



**Figure 3.3 wireless interference-01 using FriisPropagationLossModel**



**Figure 3.4 wireless interference-02 using FriisPropagationLossModel**



**Figure 3.5 wireless interference-03 using FriisPropagationLossModel**

The provided images show snapshots from a network simulation visualized in an animation tool, presumably NetAnim for ns-3. The images depict the following:

#### **(1)Nodes and Links:**

- Node 0.0 appears to be the source node, transmitting data packets.
- The other nodes, labelled from 1 to 14, seem to be in an ad hoc network, receiving data from the source.
- The blue lines from node 0.0 represent the transmission paths to the other nodes.

#### **(2)Signal Range and Signal Propagation:**

- The concentric circles around node 0.0 likely represent the signal strength as it propagates outward. Nodes closer to 0.0 will typically receive a stronger signal, while those further away will get a weaker signal.

- The signal range appears to be decreasing over time, from the first to the third image. This could represent the decay of the signal strength over distance or time, or it might be a representation of different transmission power levels at different times.

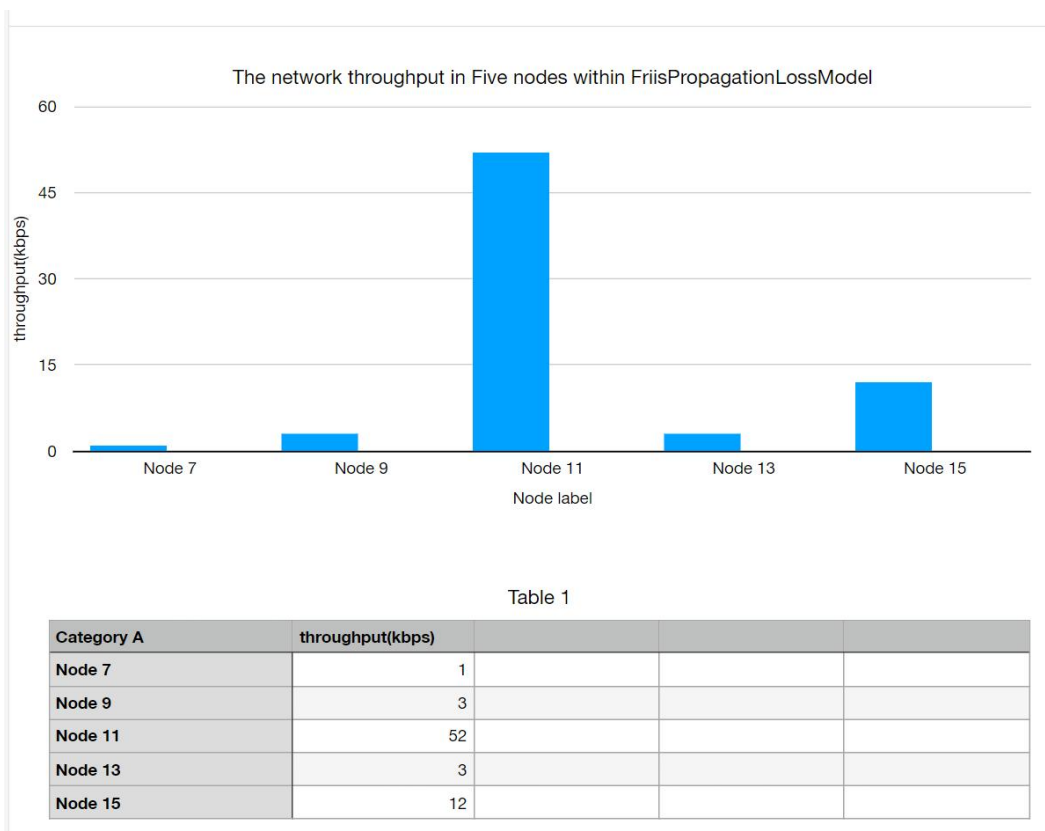
### (3)Mobility:

- It appears that there isn't significant mobility among the nodes between the different snapshots, as the positions of the nodes remain relatively stable.

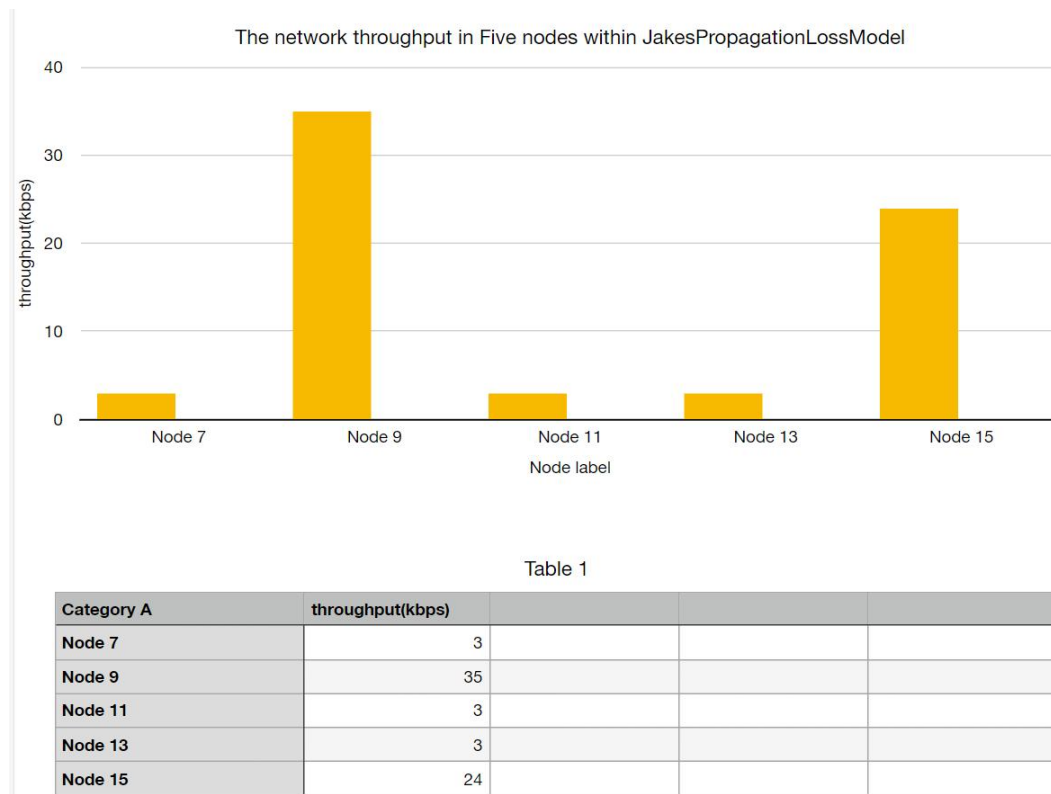
### (4)Interactions:

- The dense connections from node 0.0 to the other nodes suggest a broadcasting or a flooding mechanism where node 0.0 is sending packets to all other nodes in its range.
- Over time (from the first to the third image), fewer nodes seem to be directly connected to node 0.0, indicating that the range or the condition of the wireless medium might be changing, possibly due to factors like interference, signal attenuation, or changing transmission power.

The simulation appears to be modeling the throughput of a wireless ad hoc network under different propagation loss models FriisPropagationLossModel. The visual representation helps in understanding how signals are being propagated, which nodes are in direct communication, and potentially, how interference might be affecting network performance.



**Figure 3.6 The network throughput in Five nodes within FriisPropagationLossModel**



**Figure 3.7 The network throughput in Five nodes within JakesPropagationLossModel**

From the provided in section 3.2 data:

- **The FriisPropagationLossModel results:** showed varied throughput values: 1.20097Kbps, 2.88229Kbps, 52.1822Kbps, 2.88229Kbps, and 12.1662Kbps.
- **The JakesPropagationLossModel results:** showed throughput values: 2.88229Kbps, 35.4096Kbps, 2.88229Kbps, 2.88229Kbps, and 23.7323Kbps.

**It's evident that the throughput values fluctuate.** Some are very low, which could be a sign of high interference or other factors affecting the wireless link quality. The variation in throughput between the two models could be due to the inherent differences in how these models simulate wireless propagation, with Jakes introducing a fading component that can cause variations in signal quality over time and distance.

### 3.4 How the two models **FriisPropagationLossModel** and **JakesPropagationLossModel** are different

Both the FriisPropagationLossModel and JakesPropagationLossModel are propagation loss models used to predict signal loss over a transmission path in wireless communication, but they model different phenomena and are applicable in different scenarios.

Here's a breakdown of the differences:

#### (1)FriisPropagationLossModel:



- **Free Space Model:** It's primarily a free space propagation model, which means it's based on the concept of free-space path loss without considering obstacles or reflection.
- **Application:** It is best suited for long-range communications in an open environment without obstructions, like satellite communication.
- **Equation:** The Friis free-space equation is used to calculate path loss, which relates the received power to the transmitted power, the gain of the transmitting and receiving antennas, the distance between them, and the wavelength of the signal.
- **Loss Mechanism:** The primary factor causing signal loss in this model is the spreading of the signal over an area as it travels, often termed as path loss.

## (2) Jakes Propagation Loss Model:

- **Rayleigh Fading Model:** Jakes' model is a way to simulate Rayleigh fading, which occurs when there isn't a dominant transmission path, and there are many scatterers in the environment, leading to multipath propagation.
- **Application:** It is commonly used in densely populated urban environments where signals can reflect off buildings and other structures.
- **Equation:** It doesn't have a straightforward formula like Friis, but instead, it employs a sum of sinusoids method to simulate the effects of Rayleigh fading. This simulates the sum of multiple sinusoidal signals (reflections) with random phases, leading to a Rayleigh distribution of the resulting signal amplitude.
- **Loss Mechanism:** The primary cause of signal degradation in this model is the interference between multiple reflected copies of the signal, known as multipath fading.

### In summary:

- FriisPropagationLossModel is more about the natural loss of power a signal undergoes as it travels further away in an open space, whereas
- JakesPropagationLossModel tries to simulate the variations in signal strength over short distances due to multipath propagation, especially in environments with lots of obstacles.
- In realistic scenarios, especially in urban environments, a combination of propagation models (path loss model like Friis, fading model like Jakes, and possibly others) may be used to better represent the complexities of wireless signal propagation.

## 3.5 How interference can affect throughput:

- **(1) Reduced Data Rates:** Modern wireless communication systems, such as Wi-Fi, adjust their data rate based on the quality of the wireless link. If there's significant interference, the system might opt for a lower data rate because it's more reliable, reducing the overall throughput.

- **(2)Retransmissions:** When packets are corrupted due to interference, they have to be retransmitted. This can reduce the effective throughput because the system is spending time re-sending data rather than transmitting new data.
- **(3)Increased Latency:** Interference can lead to increased delays as the network might need to wait for a clear channel before sending data. This can especially affect real-time applications like VoIP or online gaming.
- **(4)Connection Drops:** In extreme cases, interference can lead to dropped connections, requiring devices to re-establish their link, which can interrupt data flow and reduce average throughput.
- **(5)Reduced Range:** Interference can reduce the effective range of wireless signals, limiting the areas where devices can maintain a reliable connection.

To understand the throughput values in the context of wireless interference:

- **(1)Clean Environment:** Without interference, you'd expect throughput values close to the theoretical maximum for the technology and configuration you're using.
- **(2)Moderate Interference:** With some interference, throughput might reduce somewhat due to occasional retransmissions and data rate adjustments.
- **(3)High Interference:** In a high interference environment, throughput can drop significantly. You might only achieve a fraction of the maximum potential throughput.