Part 1

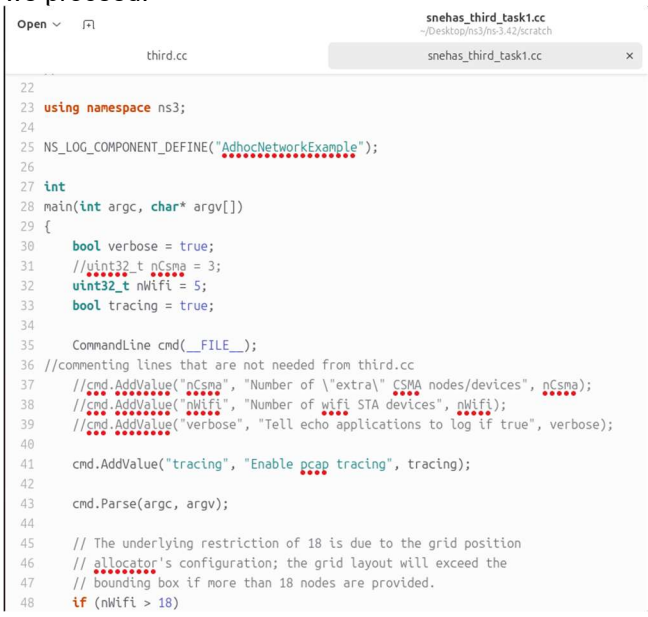
A. Experimental setup

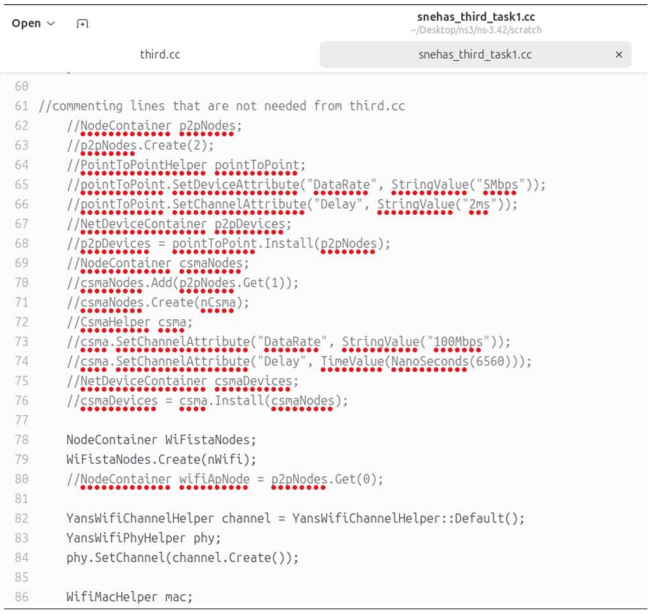
We are to define a WLAN that operates in **Ad-hoc Mode with 5 nodes**. These nodes are to move by following a 2D random walk in a rectangular area defined by the lower-left corner i.e. x = -90 m, y = -90 m and the upper-right corner i.e. x = 90 m, y = 90 m.

We take as reference and change the “third.cc” tutorial in examples in the ns3 folder, this is to ensure that my network only has WiFi nodes of **type ns3:AdhocWiFiMac.** We define the following parameters that define the features of each layer of the network model, according to the required specifications:

i. **Channel**: Default wireless channel in ns-3
ii. **Physical Layer**: Default parameters in IEEE 802.11ac standard
iii. **Link Layer**: Standard MAC without quality of service control (ad-hoc mode)
iv. **Network Layer**: IPv4 with Address range: 192.168.1.0/24
v. **Transport Layer**: UDP
vi. **Application Layer**:
   - UDP Echo Server at Node 0: Listening on port 20
   - UDP Echo Client at Node 3: Sends 2 UDP Echo packets to the server at times 1s and 2s
   - UDP Echo Client at Node 4: Sends 2 UDP Echo packets to the server at times 2s and 4s
   - Packet size: 512 bytes
vii. **Additional parameters**: Set up a packet tracer ONLY on Node 1

Below attached are screenshots of the changes made in the code accompanied by the comments. Since we are editing a copy of the third.cc file to incorporate our changes, there are few lines that are to be //commented as we proceed:





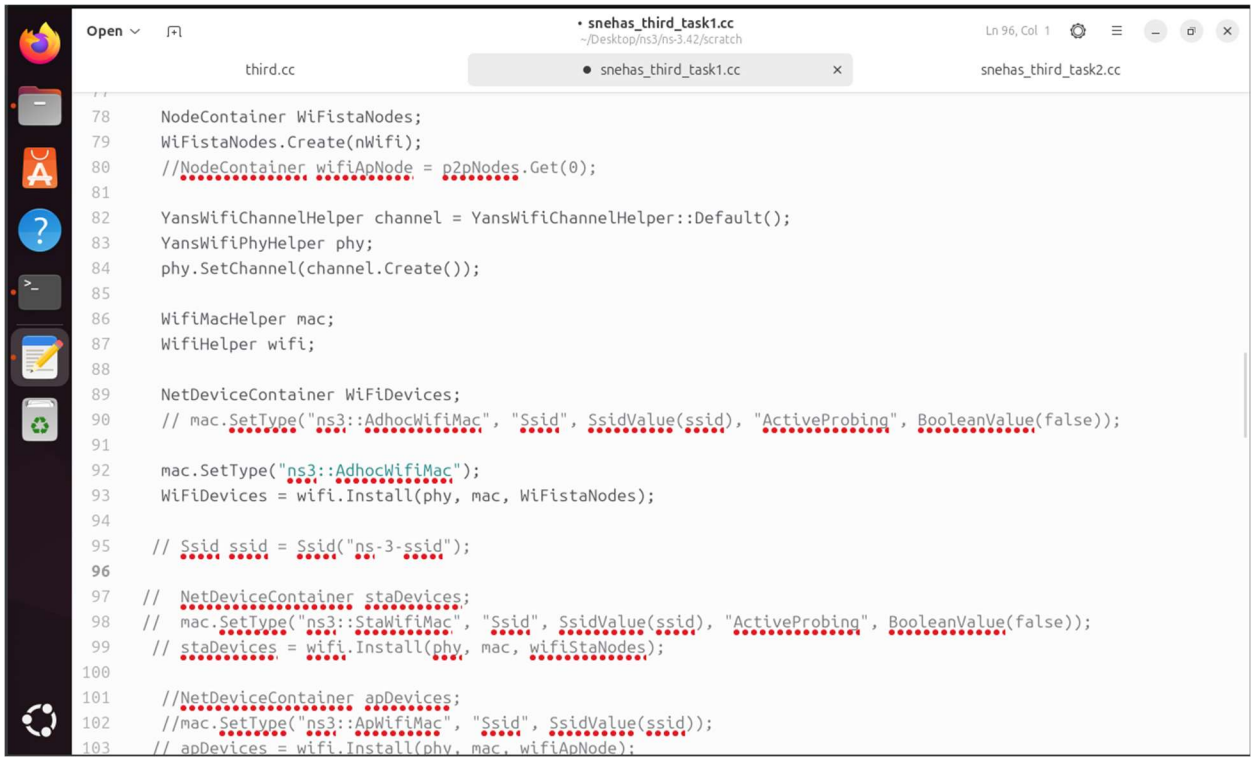


Fig. 1.1. Activating the Adhoc Wifi Mode

Here in line #92, we set the type of network connection to Adhoc Wifi. The **AdhocWifiMac** setting is essential for configuring the nodes to operate in Ad-Hoc Mode rather than Infrastructure Mode, which is typically used for Wi-Fi networks involving an Access Point (AP) and stations (STA). This allows them to communicate directly without relying on any AP. This setting is fundamental for simulating scenarios where nodes need to interact as peers, such as in mesh networks, sensor networks, and other decentralized network setups.

Fig. 1.2. Define the corner points of the mobility rectangle in line #125

Fig. 1.3. Modifications highlighted by comments

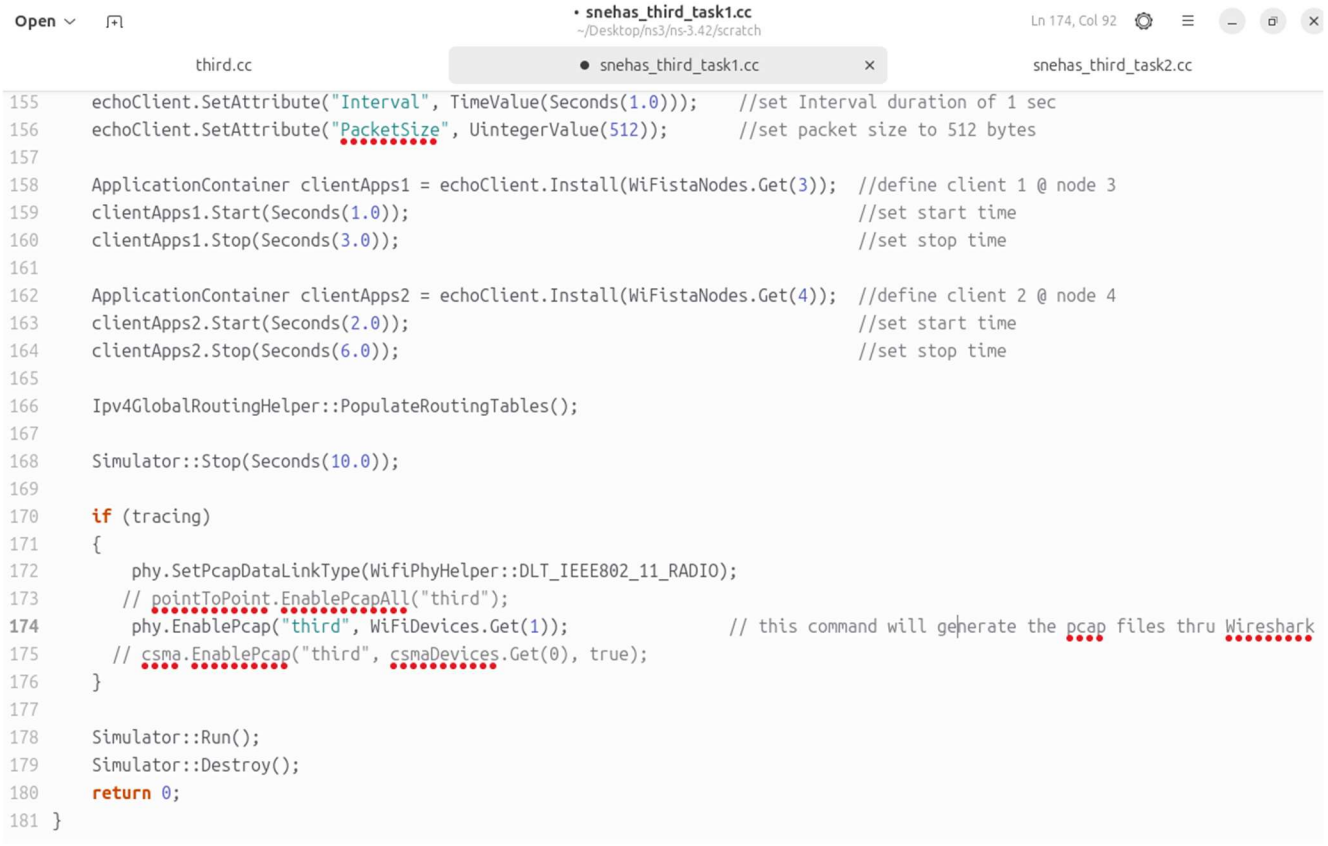


Fig. 1.4. Modifications highlighted by comments

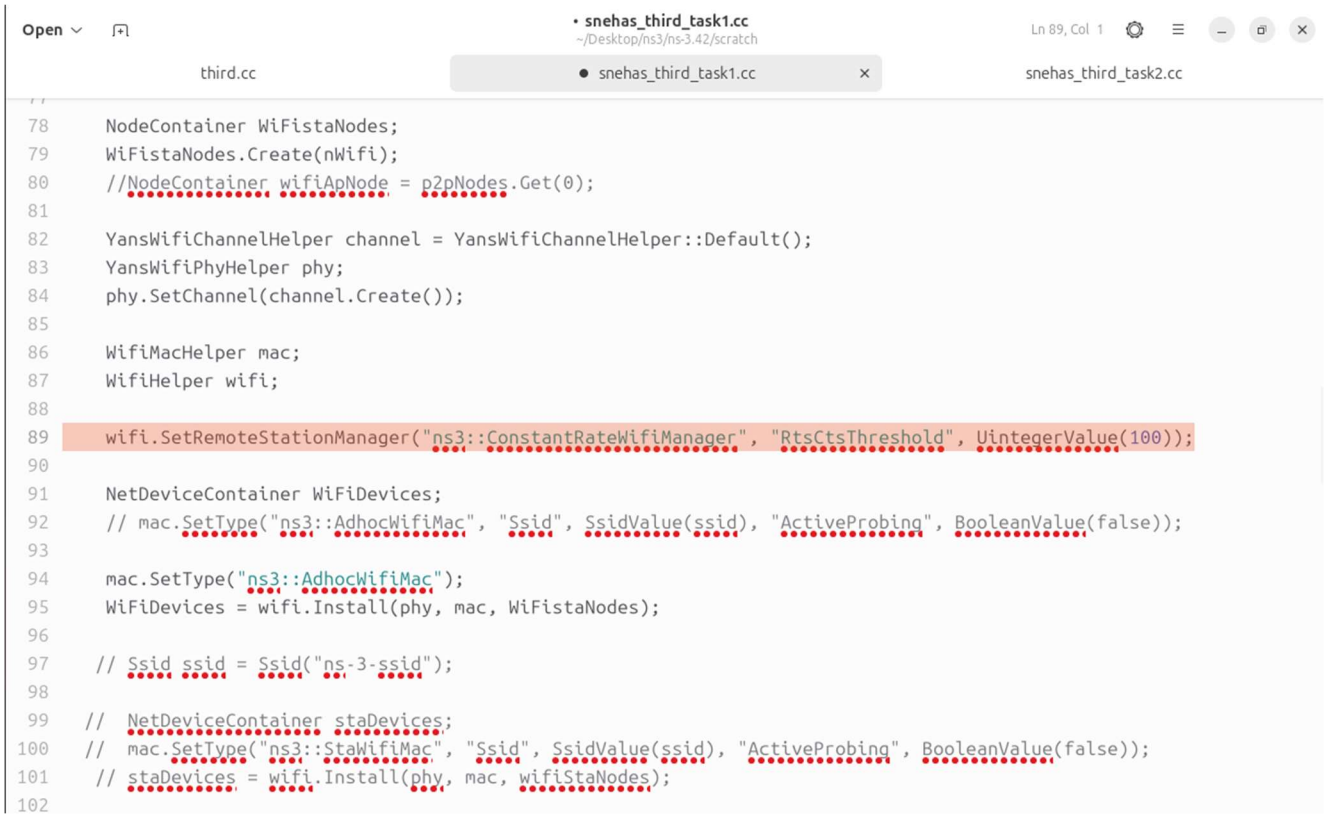


Fig. 1.5. Forcing the utilization of RTS/CTS in the AdHoc network

The RTS/CTS (Request to Send / Clear to Send) handshake can be enabled in ns-3 to reduce collisions, particularly in scenarios with potential hidden node issues.

The RTS/CTS mechanism can be enabled by setting the RTS threshold in ns3::WifiRemoteStationManager. By configuring the RtsCtsThreshold attribute to 0, every data frame in the network will trigger an RTS/CTS handshake, regardless of its size.

Set the RTS Threshold:

- RTS/CTS is typically triggered when the packet size exceeds a certain threshold.
- By setting the **RTS threshold to 0 bytes**, you can force every data frame to go through the RTS/CTS handshake, regardless of its size.

Modify the ns-3 Code to Set the RTS Threshold:

- In your code, add the following line before installing devices on nodes:

```
wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager", "RtsCtsThreshold", UintegerValue(100));
```

This line sets the RTS/CTS threshold to 0 bytes, ensuring that all data frames will use RTS/CTS.

Explanation of RTS/CTS in Wi-Fi:

- RTS/CTS helps to avoid collisions by "reserving" the channel for a particular node before it transmits data.
- **RTS** (Request to Send) is sent by the sender to inform other nodes about its intent to send data.
- The receiving node replies with **CTS** (Clear to Send), allowing the sender to transmit data. Other nodes "overhear" the CTS and avoid sending data, thus reducing the chance of collisions.

B. Results

Following are snapshots of the output terminal for the above set up:

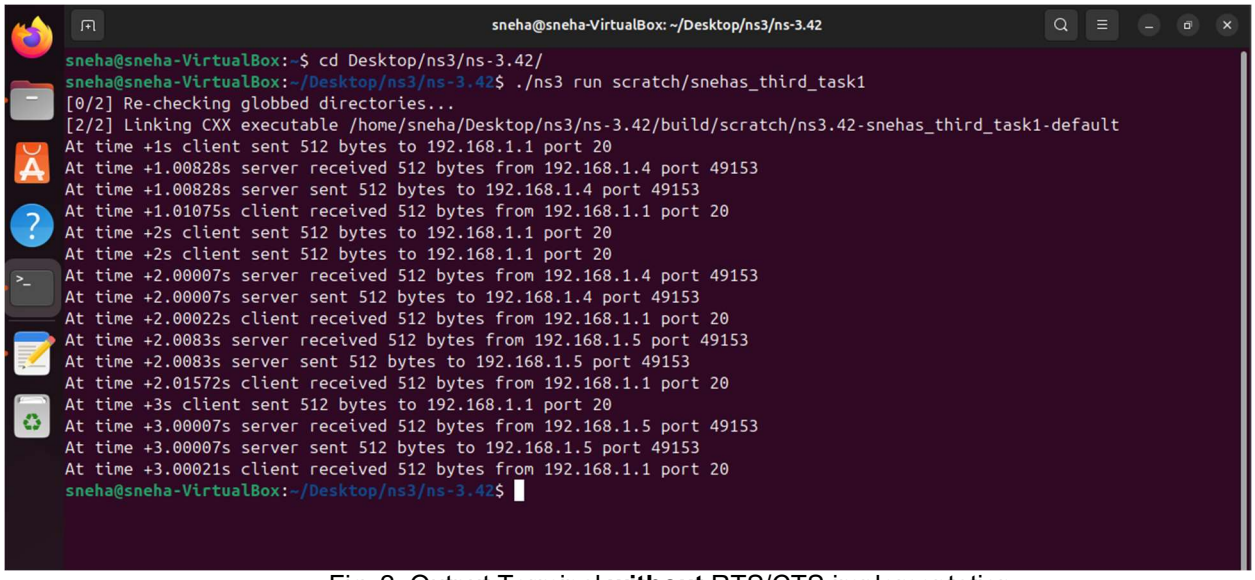


Fig. 2. Output Terminal **without** RTS/CTS implementation

We also look at the pcap files generated in Wireshark:

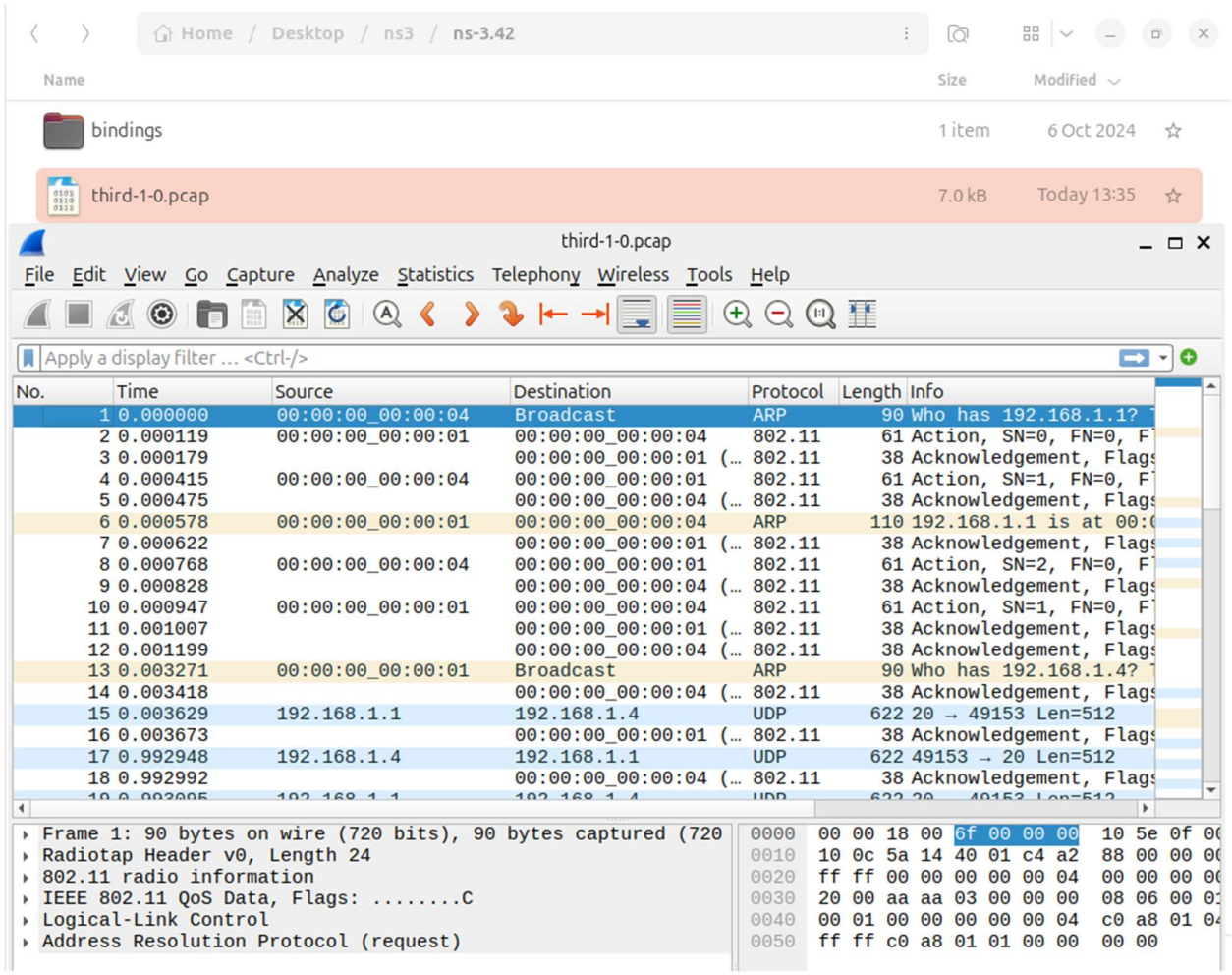


Fig. 3. Wireshark pcap **without** RTS/CTS implementation

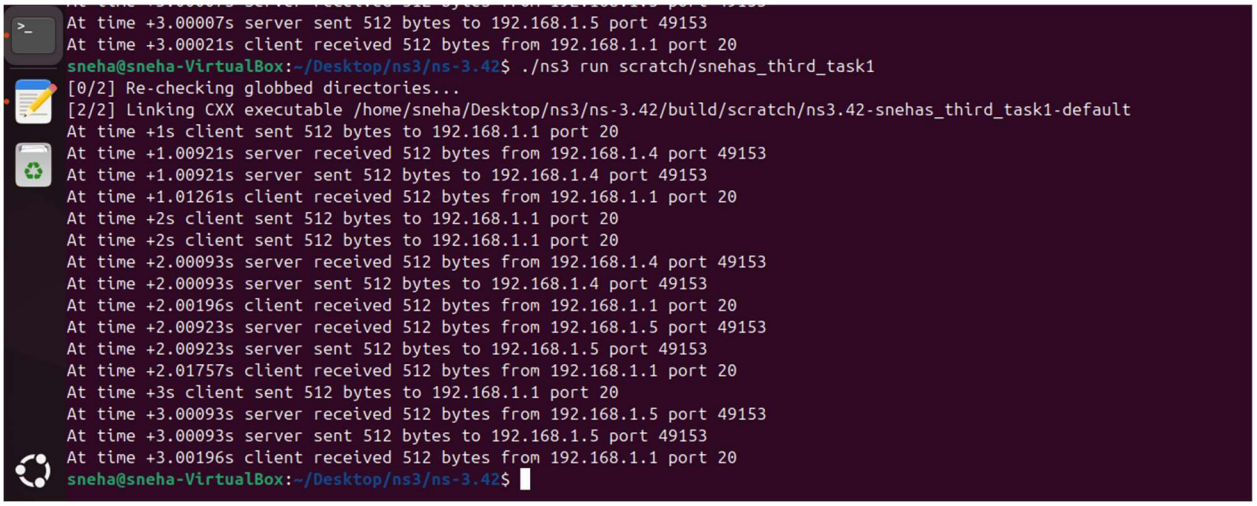


Fig. 4. Output Terminal **with** RTS/CTS implementation

Here, we notice a slight delay in milliseconds for each of the timestamps, as compared to the previous output terminal window. This is because the RTS/CTS handshake requires extra steps to ensure the channel is clear before transmitting data. The primary benefit of RTS/CTS is that it reduces collisions, especially in environments with hidden nodes. By reserving the channel before sending large data frames, RTS/CTS minimizes the likelihood of simultaneous transmissions and subsequent data loss.

The RTS/CTS handshake includes scanning the channel to check if it's idle. If the channel is occupied, the sender waits and retries until the channel is free.This two-step handshake takes time, as the sender and receiver must wait for each frame to be acknowledged and the channel to be cleared.

Even when the channel is clear, the RTS and CTS frames introduce propagation and processing delay. The time taken to transmit these frames and process the responses adds a few milliseconds of delay before the actual data frame is transmitted.

In the pcap files, we can see the “Request-to-send” and “Clear-to-send” frames, that is how we know it has been implemented successfully.

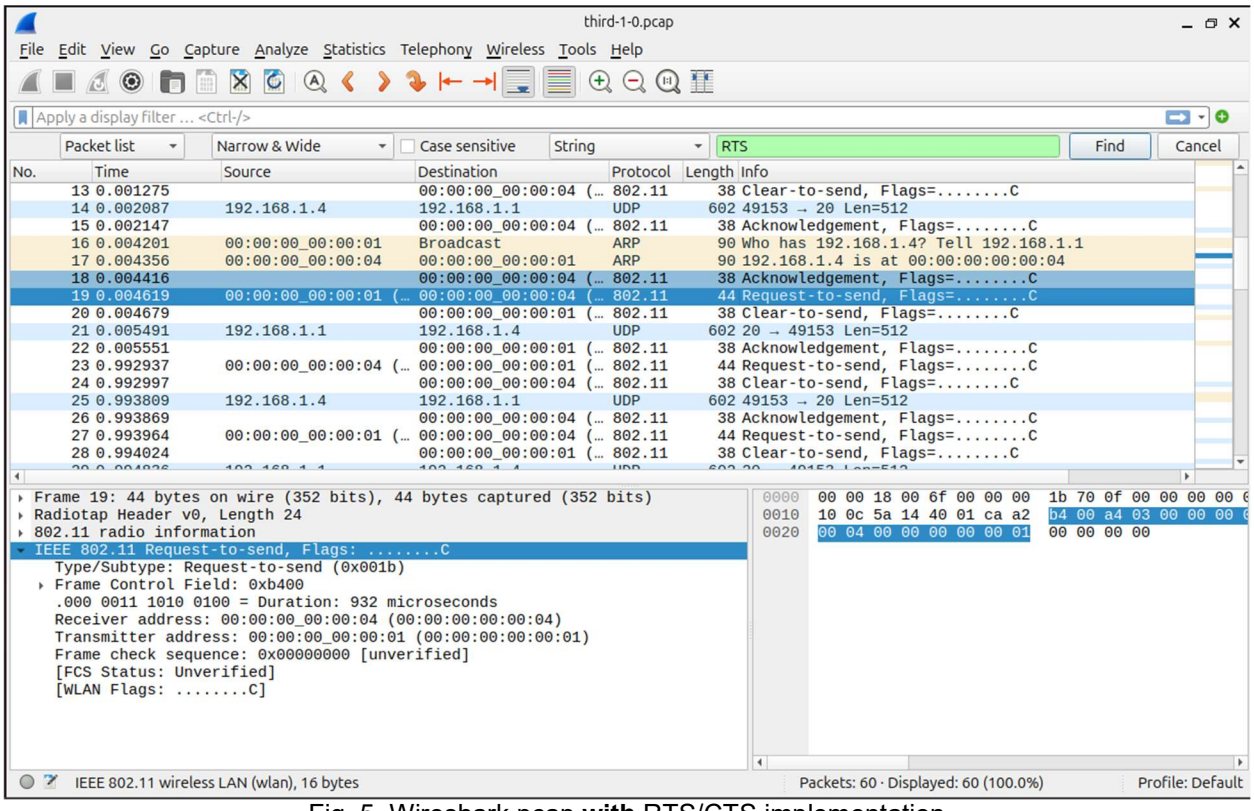


Fig. 5. Wireshark pcap **with** RTS/CTS implementation

Now let's answer the following questions:

a) Are all the frames acknowledged? Explain why.

- To determine if all frames are acknowledged, we can analyze the captured .pcap file for any signs of unacknowledged frames. In Wi-Fi ad-hoc mode, frames are usually acknowledged at the MAC layer, so each data frame sent by one node should be acknowledged by the receiving node.
- Acknowledgments (ACK frames) are sent by the receiver to confirm the successful receipt of dataframes. This implies that acknowledgments are enabled in the network configuration. In the standard setup described, where nodes operate in ad-hoc mode with a standard MAC, acknowledgments are typically enabled by default.
- To verify, we look for pairs of data and ACK frames in the .pcap file for each unicast packet. For each data frame, there should be a corresponding ACK frame. Here, each data frame has an ACK, thus all frames are acknowledged.
-

b) Are there any collisions in data frames? Explain why. How have you reached this conclusion?

- To determine if there are any collisions in data frames, you can analyze the .pcap file for signs of retransmissions or missing ACKs, which are indicators of collisions. In Wi-Fi networks, if a frame experiences a collision, it is typically retransmitted by the sender. We identify and look for retransmissions in the pcap file.
- For each data frame sent, an ACK is expected in a short time window. If an ACK is missing, it may indicate that the original frame encountered a collision. Missing ACKs without a retry could also mean that the frame was lost due to interference or that the destination node was out of range.

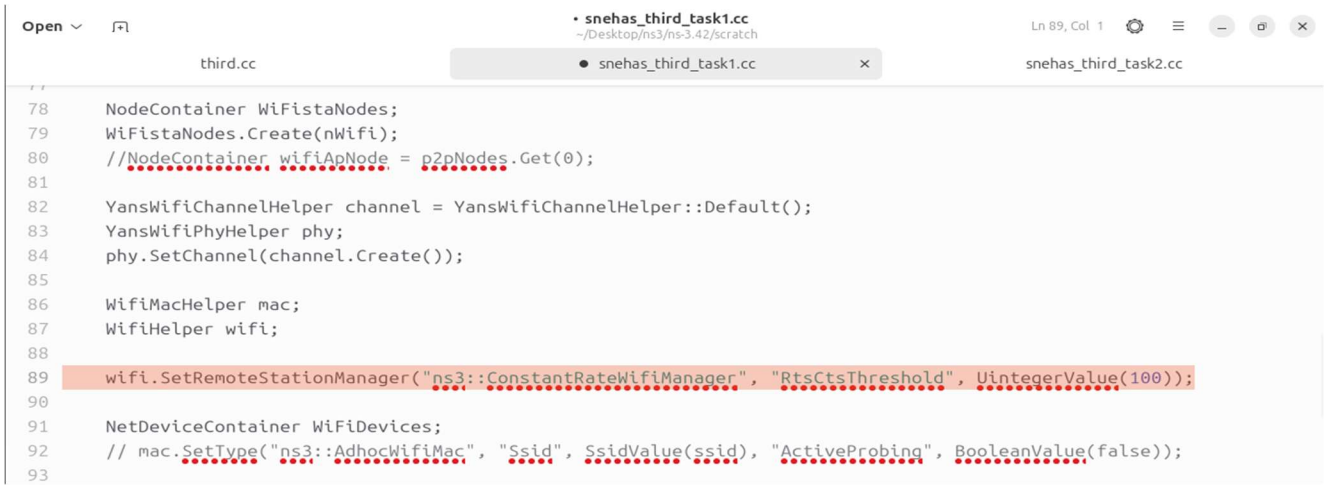
  - **Why Collision Happens?** In an ad-hoc Wi-Fi network, there is no central coordination (like an access point) to manage traffic. Therefore, nodes transmitting simultaneously within range of each other can cause collisions. Additionally, hidden node problems (nodes that can't detect each other's transmissions) in ad-hoc networks can increase collision likelihood.
  - **How to conclude collisions presence**: By examining the .pcap file in Wireshark with the filter mentioned, if retransmissions are observed, this directly indicates collisions. Alternatively, missing ACKs without retry attempts could also suggest issues like interference or range limitations.
    - If you see **retransmissions** (frames with the retry bit set) or **missing ACKs**, then **yes, there are collisions**.
    - If no retransmissions are observed and each data frame has an ACK, it would suggest that **no collisions occurred**.

c) How can the nodes be forced to utilize the RTS/CTS handshake procedure?

- The RTS/CTS (Request to Send / Clear to Send) handshake can be enabled in ns-3 to reduce collisions, particularly in scenarios with potential hidden node issues. Here's how you can force nodes to use the RTS/CTS mechanism:

```
wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager", "RtsCtsThreshold", UintegerValue(100));
```

Here's how we included it in the code:



- With the “ns3::ConstantRateWifiManager” set as the Remote Station Manager, and the RTS/CTS set to a threshold of 100 bytes, the RTS/CTS handshake will be triggered when the packet size exceeds 100 bytes. This means that if a node wants to send a packet larger than 100 bytes, it will first send an RTS frame to request permission to send. If the receiver hears the RTS and is not currently busy, it responds with a CTS frame, indicating that it's clear to send.

- RTS/CTS helps to avoid collisions by "reserving" the channel for a particular node before it transmits data.

- **RTS** (Request to Send) is sent by the sender to inform other nodes about its intent to send data.The receiving node replies with **CTS** (Clear to Send), allowing the sender to transmit data. Other nodes "overhear" the CTS and avoid sending data, thus reducing the chance of collisions.

d) Force the utilization of RTS/CTS in the network:

- **Are there any collisions in data frames?**
- After enabling RTS/CTS, you can observe the .pcap file to check for retransmissions, which indicate collisions. The RTS/CTS mechanism should reduce or eliminate collisions, especially in scenarios with hidden nodes, by reserving the channel before data transmission. We see no retransmitted frames, it suggests that RTS/CTS is effectively reducing collisions.
- **Which is the benefit or RTS/CTS? Briefly explain how RTS/CTS works.**
- The primary benefit of RTS/CTS is to reduce collisions, particularly in networks with hidden nodes. In ad-hoc or dense environments, collisions are more likely because nodes may not be aware of each other's transmissions due to distance or obstacles. RTS/CTS helps to reserve the channel for a sender, preventing other nodes from transmitting simultaneously and causing a collision.

How RTS/CTS Works

- RTS (Request to Send): The sender first sends an RTS frame to the receiver, indicating its intent to transmit data.
- CTS (Clear to Send): If the receiver is ready and the channel is clear, it responds with a CTS frame
- Data Transmission: Upon receiving the CTS, the sender proceeds with the actual data transmission.
- Acknowledgment (ACK): Once the data is successfully received, the receiver sends an ACK to the sender.
- This handshake effectively "reserves" the channel for the sender, informing other nodes to avoid transmitting until the communication completes.

- **Which is the benefit or RTS/CTS? Briefly explain how RTS/CTS works.**
- The Network Allocation Vector (NAV) information is maintained at the Physical (PHY) layer of the IEEE 802.11 standard. In ns-3, which models this standard, you can access and manipulate the NAV using the various attributes and methods provided by the WifiPhy class. Specifically, you can find the NAV information within the WifiPhy module of ns-3. The WifiPhy class represents the physical layer characteristics of the WiFi protocol stack. It includes attributes and methods related to various aspects of PHY layer operation, including NAV.
- The NAV is a timer that is maintained by each node in a wireless network. It represents the duration of time for which the medium is reserved. The NAV is set based on the information contained in the duration field of received frames, such as RTS (Request to Send) and CTS (Clear to Send) frames. When a node receives an RTS or CTS frame, it updates its NAV based on the duration information in that frame. During this time period, the node refrains from attempting to transmit data, as the channel is reserved for the ongoing transmission.

So, in a way, we can think of the NAV as an extension of the information provided in the duration field. The duration field is specific to a single frame, while the NAV represents a time interval during which the channel is reserved based on the cumulative information received in various frames. In summary, the NAV is closely related to the duration field, as it is derived from the information contained in the duration field of received frames. They work together to manage the shared wireless medium and avoid collisions in a wireless network.

C. Learnt Lessons

Task 1 teaches lessons about managing collisions, configuring ad-hoc networks, and using tools like Wireshark for network analysis. Understanding these concepts is crucial for building efficient, decentralized networks in real-world applications like IoT and mesh networks. Here are some key lessons learned from this task:

1. Understanding Ad-Hoc Mode in Wi-Fi Networks

- **Direct Communication**: Ad-hoc mode allows devices (nodes) to communicate directly without relying on an access point (AP). This is suitable for decentralized networks, such as peer-to-peer setups and sensor networks.
- **Peer-to-Peer Configuration**: Unlike infrastructure mode, each node in an ad-hoc network can dynamically establish connections with nearby nodes, which is useful for scenarios where centralized coordination is unnecessary or impractical.

2. Collision Management and the RTS/CTS Mechanism

- **Challenges with Collisions**: In ad-hoc networks, collisions are more likely because there's no AP to manage access to the medium. Nodes can transmit simultaneously, causing collisions, especially in environments with hidden nodes.
- **Using RTS/CTS to Reduce Collisions**: By forcing the RTS/CTS handshake, you can reduce collisions. RTS/CTS is particularly beneficial in ad-hoc networks, as it helps to avoid hidden node issues and creates a form of virtual carrier sensing.
- **Impact of RTS/CTS Threshold**: Setting the RTS/CTS Threshold to zero ensures that every packet triggers RTS/CTS, which, while effective at reducing collisions, can also increase overhead. Finding the right balance is crucial for performance in real-world scenarios.

3. Network Allocation Vector (NAV) and Channel Reservation

- **Role of NAV**: NAV is a virtual carrier sensing mechanism that allows nodes to "reserve" the medium by specifying the duration for which they will occupy it. This prevents other nodes from transmitting during this period, reducing the likelihood of collisions.
- **Using NAV to Analyze Network Performance**: Monitoring NAV values in Wireshark helps in understanding how nodes avoid collisions, as nodes respect the NAV duration set by other nodes' RTS/CTS and data frames.

4. Packet Analysis and Interpretation in Wireshark

- **Identifying Acknowledgments**: Checking whether frames are acknowledged helps verify reliable communication. Every data frame in ad-hoc mode should ideally have a corresponding ACK, indicating successful delivery.
- **Detecting Retransmissions**: By examining retransmitted frames, you can identify collision points or lost packets. This analysis is essential for understanding network performance and the effectiveness of collision control mechanisms like RTS/CTS.

5. Configuring ns-3 for Realistic Network Behavior

- **Using Config::SetDefault for Global Settings**: ns-3's Config::SetDefault function lets you globally apply settings like the RtsCtsThreshold, ensuring consistent behavior across all nodes. This simplifies network-wide configuration.
- **Setting Up Mobility Models**: The random walk mobility model provides realistic movement for nodes in an ad-hoc network. Movement patterns are important in ad-hoc networks, as nodes may frequently move in and out of each other's range.

6. Trade-Offs and Limitations

- **Overhead of RTS/CTS**: While RTS/CTS reduces collisions, it also adds overhead, as every transmission must initiate an RTS/CTS handshake. In a dense or high-traffic network, this can impact overall throughput and latency.
- **Balancing Reliability and Performance**: In network configurations, there's often a trade-off between reducing collisions (using RTS/CTS) and maintaining high throughput. Real-world ad-hoc networks may adjust the RTS/CTS threshold dynamically to balance this trade-off.

Part 2

A. Experimental setup

We are to define a WLAN that operates in **Infrastructure Mode with 5 nodes and 1 access point**. These nodes are to move by following a 2D random walk in a rectangular area defined by the lower-left corner i.e. x = -90 m, y = -90 m and the upper-right corner i.e. x = 90 m, y = 90 m.

The access point remains in a fixed position. The network name (SSID) should be EECE5155.

i. **Channel**: Default wireless channel in ns-3
ii. **Physical Layer**: Default parameters in IEEE 802.11ac standard
iii. **Link Layer**: Standard MAC without quality of service control (infrastructure mode)
iv. **Network Layer**: IPv4 with Address range: 192.168.2.0/24
v. **Transport Layer**: UDP
vi. **Application Layer**:
    - UDP Echo Server at Node 0: Listening on port 21
    - UDP Echo Client at Node 3: Sends 2 UDP Echo packets to the server at times 3s and 5s
    - UDP Echo Client at Node 4: Sends 2 UDP Echo packets to the server at times 2s and 5s
    - Packet size: 512 bytes
vii. **Additional parameters**: Set up a packet tracer ONLY on Node 4 and on AP (Node 5 in my case)

Below attached are screenshots of the changes made in the code accompanied by the comments. We talk about the newer modifications needed for Task 2:

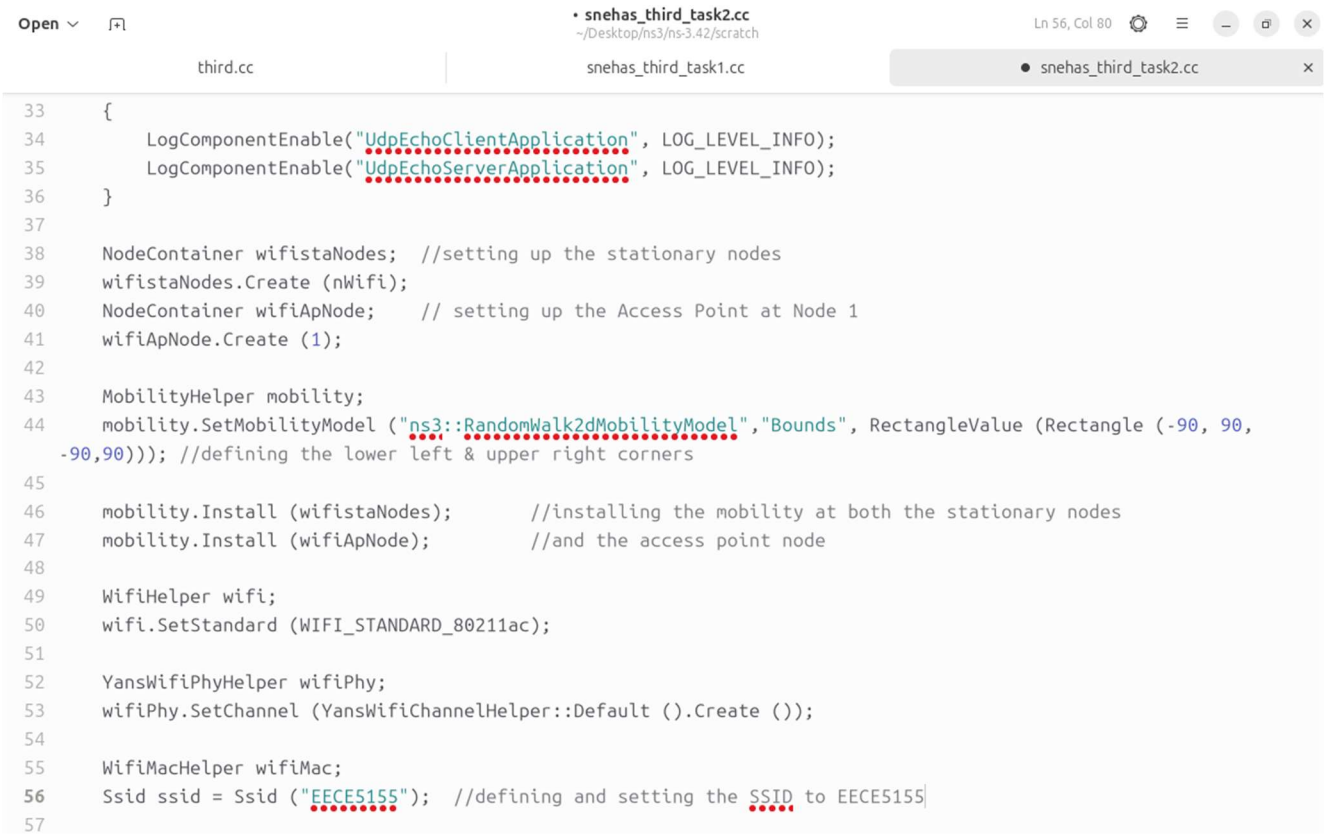


Fig. 6.1. Modifications highlighted by comments

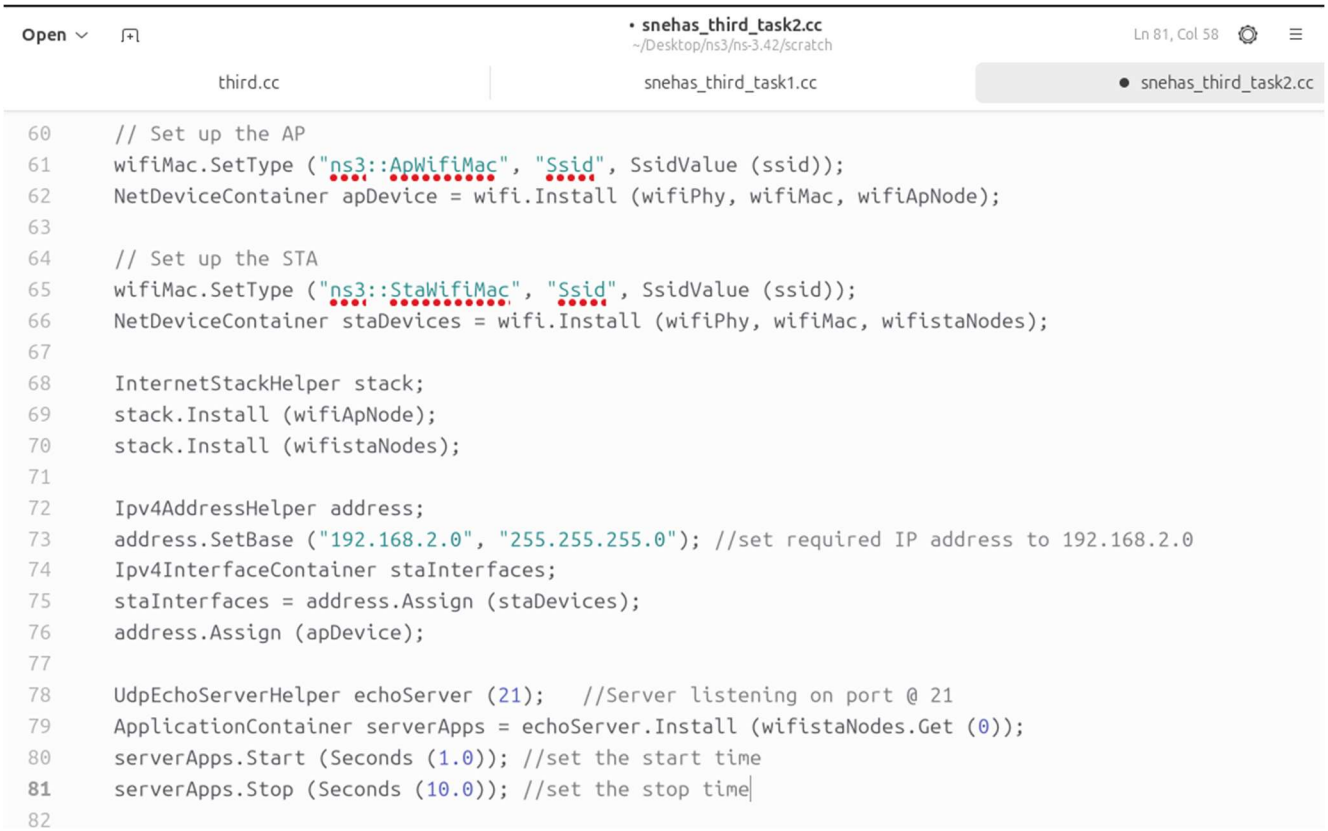


Fig. 6.2. Modifications highlighted by comments

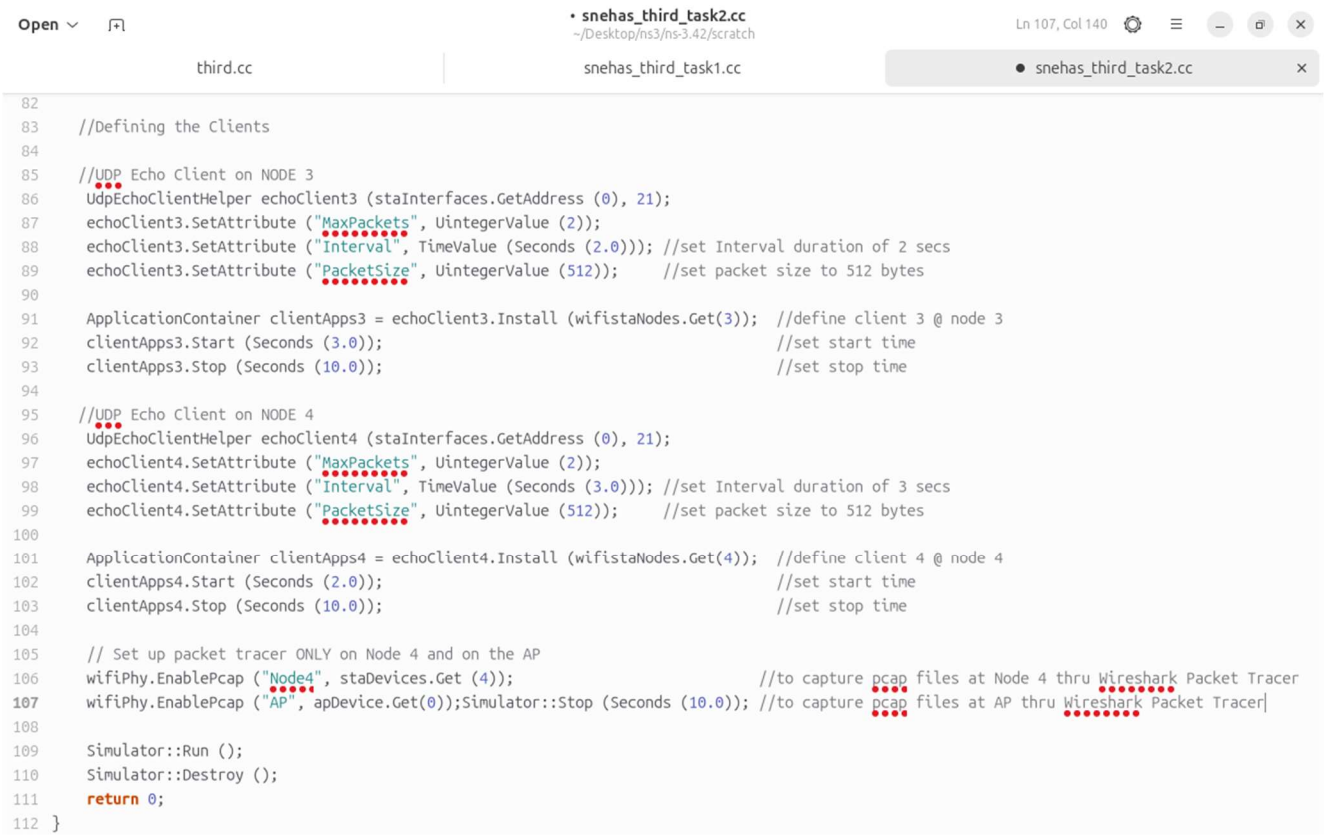


Fig. 6.3. Modifications highlighted by comments

As in task 1, in the similar way we define the Remote Station Manager here in line #58.

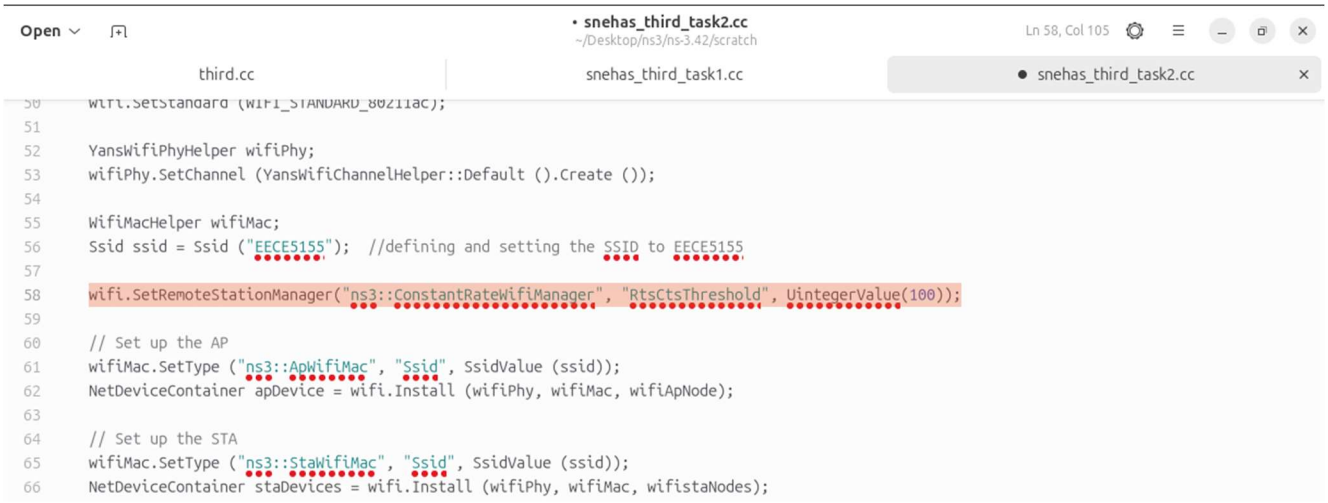


Fig. 7. Forcing the utilization of RTS/CTS in the Infrastructure Mode

B. Results

Following are snapshots of the output terminal for the above set up:

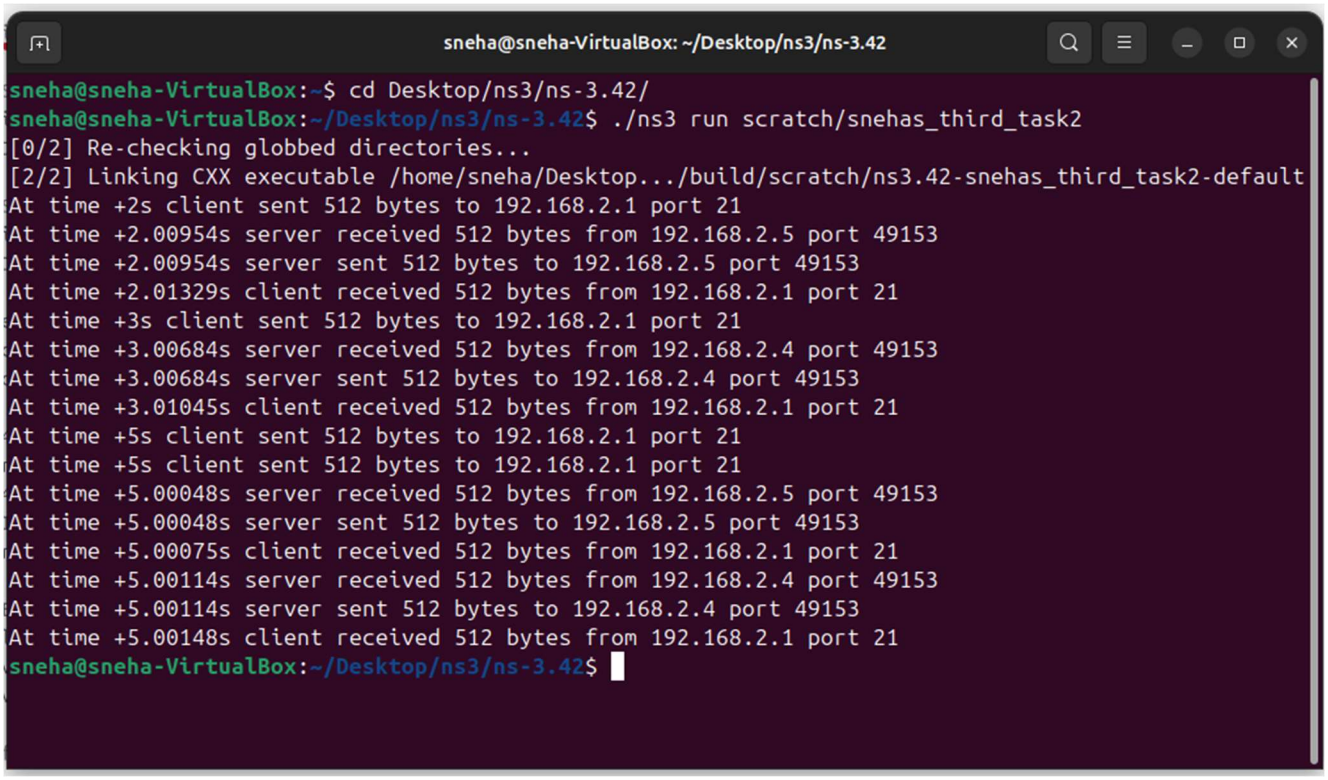


Fig. 8. Output Terminal **without** RTS/CTS implementation in Infrastructure mode

Observing the pcap files in Wireshark tabs at both Node 4 and Access Point (Node 5):

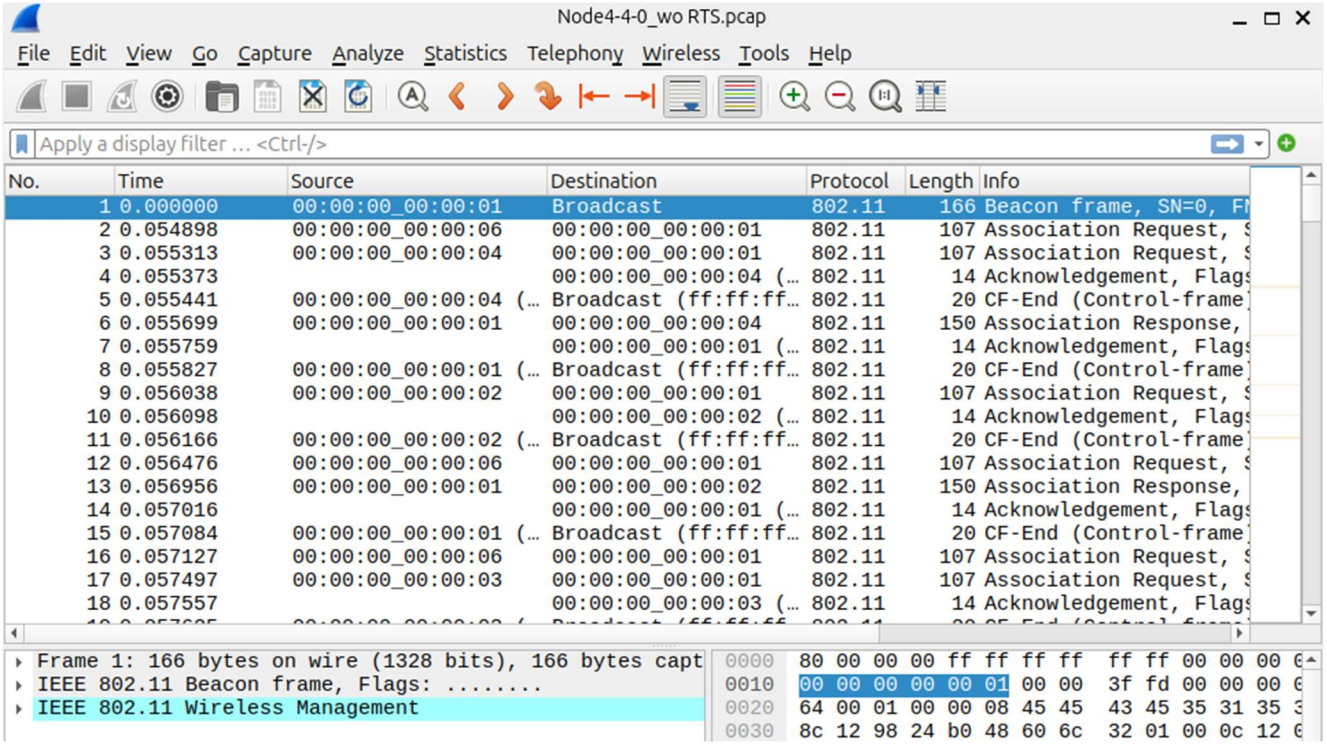


Fig. 9. Packet Capture at **Node 4 without** RTS/CTS implementation in Infrastructure mode

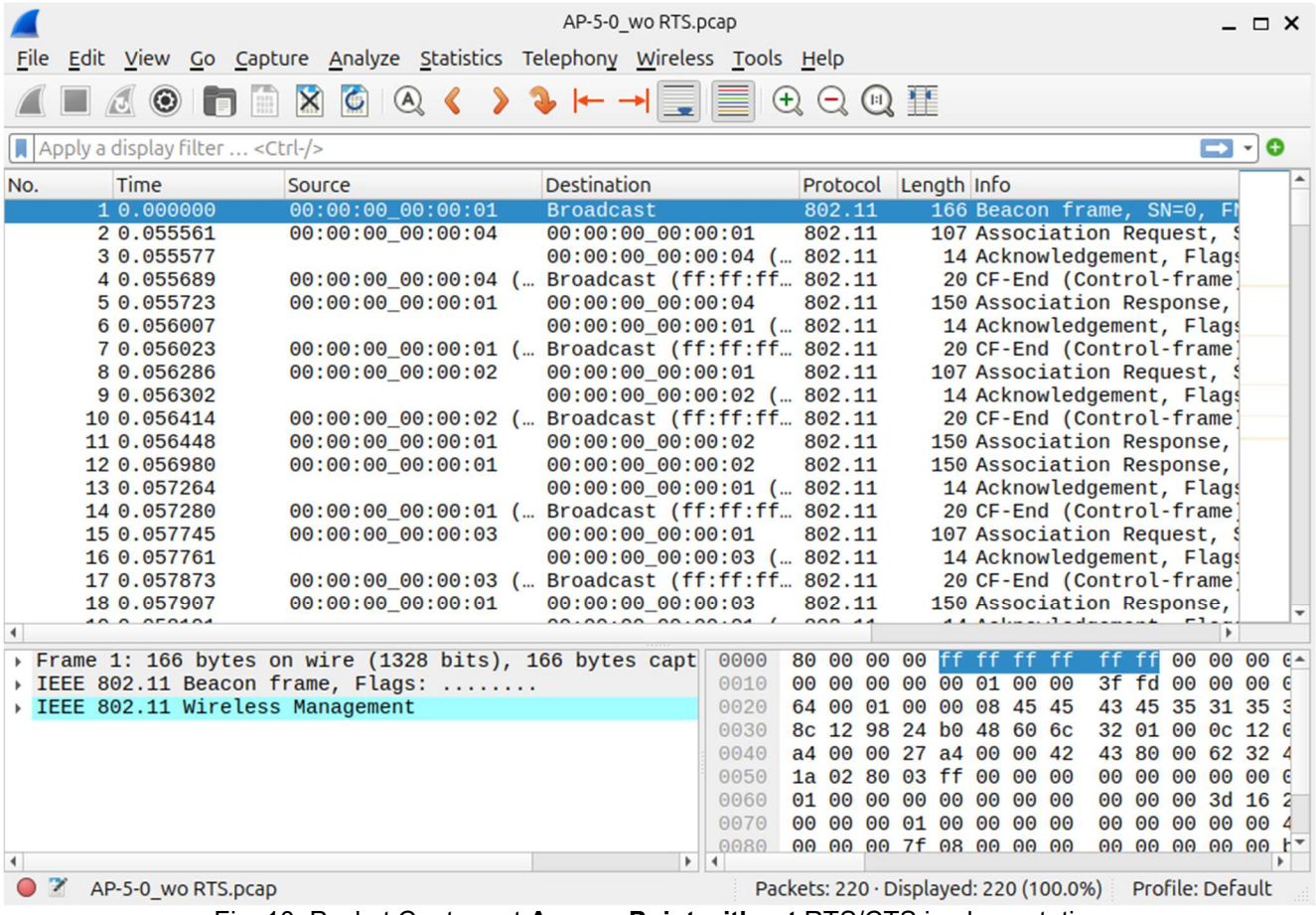


Fig. 10. Packet Capture at **Access Point without** RTS/CTS implementation

```
sneha@sneha-VirtualBox:~/Desktop/ns3/ns-3.42$ ./ns3 run scratch/snehas_third_task2
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable /home/sneha/Desktop.../build/scratch/ns3.42-snehas_third_task2-default
At time +2s client sent 512 bytes to 192.168.2.1 port 21
At time +2.01185s server received 512 bytes from 192.168.2.5 port 49153
At time +2.01185s server sent 512 bytes to 192.168.2.5 port 49153
At time +2.01763s client received 512 bytes from 192.168.2.1 port 21
At time +3s client sent 512 bytes to 192.168.2.1 port 21
At time +3.00887s server received 512 bytes from 192.168.2.4 port 49153
At time +3.00887s server sent 512 bytes to 192.168.2.4 port 49153
At time +3.01478s client received 512 bytes from 192.168.2.1 port 21
At time +5s client sent 512 bytes to 192.168.2.1 port 21
At time +5s client sent 512 bytes to 192.168.2.1 port 21
At time +5.00225s server received 512 bytes from 192.168.2.5 port 49153
At time +5.00225s server sent 512 bytes to 192.168.2.5 port 49153
At time +5.0043s client received 512 bytes from 192.168.2.1 port 21
At time +5.00648s server received 512 bytes from 192.168.2.4 port 49153
At time +5.00648s server sent 512 bytes to 192.168.2.4 port 49153
At time +5.00861s client received 512 bytes from 192.168.2.1 port 21
sneha@sneha-VirtualBox:~/Desktop/ns3/ns-3.42$
```

Fig. 11. Output Terminal **with** RTS/CTS implementation in Infrastructure mode

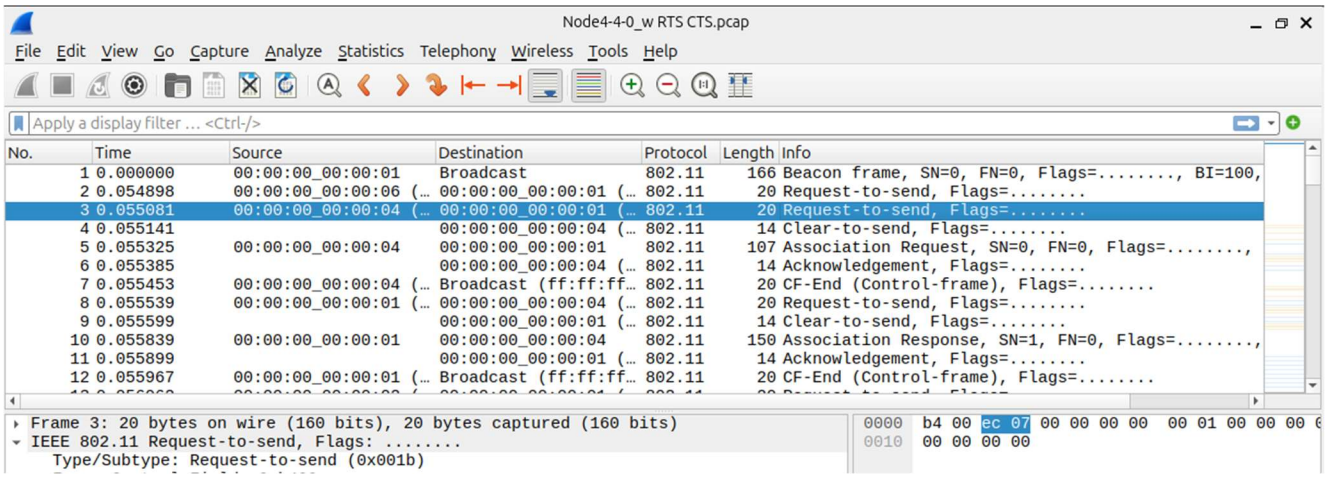


Fig. 12. Packet Capture at Node 4 **with** RTS/CTS implementation

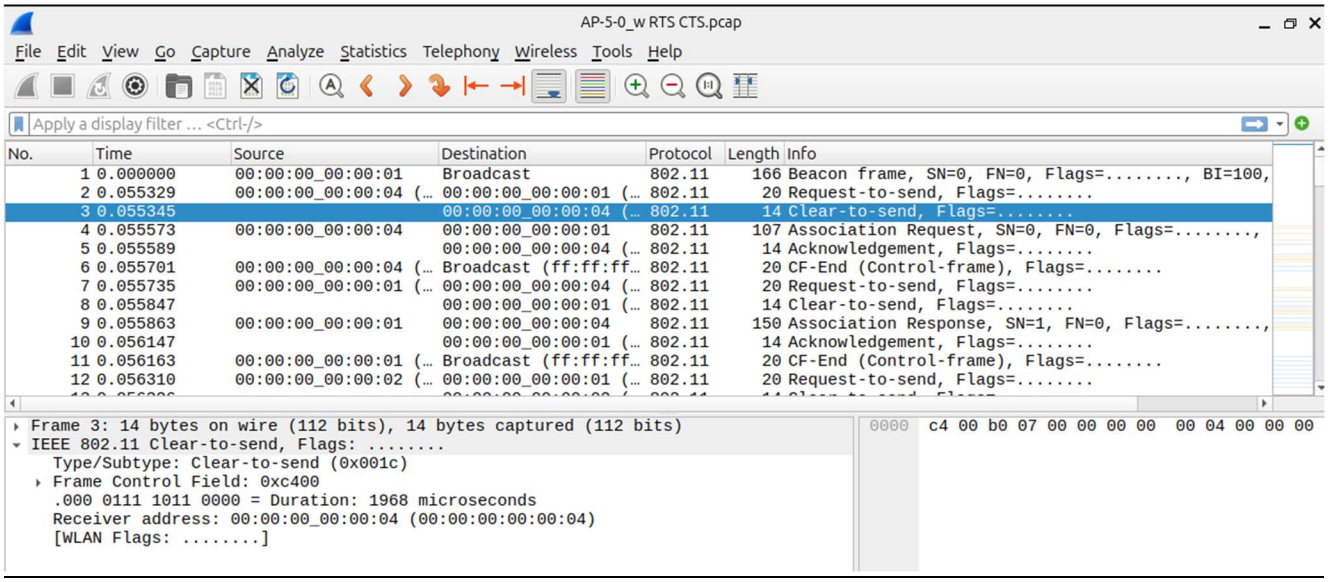


Fig. 13. Packet Capture at Access Point **with** RTS/CTS implementation

Now let's answer the following questions:

a) Explain the behavior of the AP. What is happening since the very first moment the network starts operating?

- The AP operates as a central coordinator for the Wi-Fi network, providing connectivity, traffic management, and communication between STA nodes. It initiates its role by broadcasting its SSID and then continuously manages and routes traffic in the network.
- **Access Point Initialization:**
  - The AP is created as a separate node (wifiApNode) with a specified SSID (EECE5155), which serves as the identifier for the Wi-Fi network.
  - The AP is configured to operate on the WIFI_STANDARD_80211ac protocol, providing high-speed Wi-Fi access.
  - The AP is set up with an ApWifiMac type, indicating its role as an Access Point, allowing it to manage communication with connected stations (STA).
- **Channel Setup and Initial Broadcasting:**
  - When the network starts, the AP broadcasts its SSID (EECE5155) over the Wi-Fi channel, allowing other devices (STA nodes) to detect its presence.
  - Stations in range scan for available networks and recognize the AP due to the broadcasted SSID. This is how the STA nodes discover and attempt to connect to the AP**.**
- **IP Address Assignment:**
  - Once the AP and STA nodes are initialized, the InternetStackHelper assigns IP addresses to both the AP and the STA nodes in the 192.168.2.0 subnet. This allows the nodes to communicate over the network layer.
  - The AP is assigned an IP address, allowing it to serve as a central hub for routing traffic between STA nodes and any external clients.
- **Traffic Management and Routing:**
  - The AP acts as a router for the connected STA nodes, managing their traffic by forwarding packets to the appropriate destination. It receives and forwards packets sent from the STA nodes, ensuring they reach their intended recipients.
- **Handling UDP Echo Server Requests:**
  - The AP does not directly serve as the UDP server in this setup, as one of the STA nodes (Node 0) is configured as the UDP Echo Server.
  - The AP routes packets between the UDP clients (Node 3 and Node 4) and the server (Node 0). It helps manage the flow of UDP packets, including forwarding packets from the clients to the server and vice versa.
- **Packet Capturing and Analysis:**
  - Packet tracing is enabled on both the AP and Node 4. This allows monitoring of packets that pass through the AP, capturing data for network analysis or debugging.
  - The .pcap files generated can be analyzed to observe the AP's interactions with the STA nodes and check if it efficiently manages packet routing without collisions or loss.

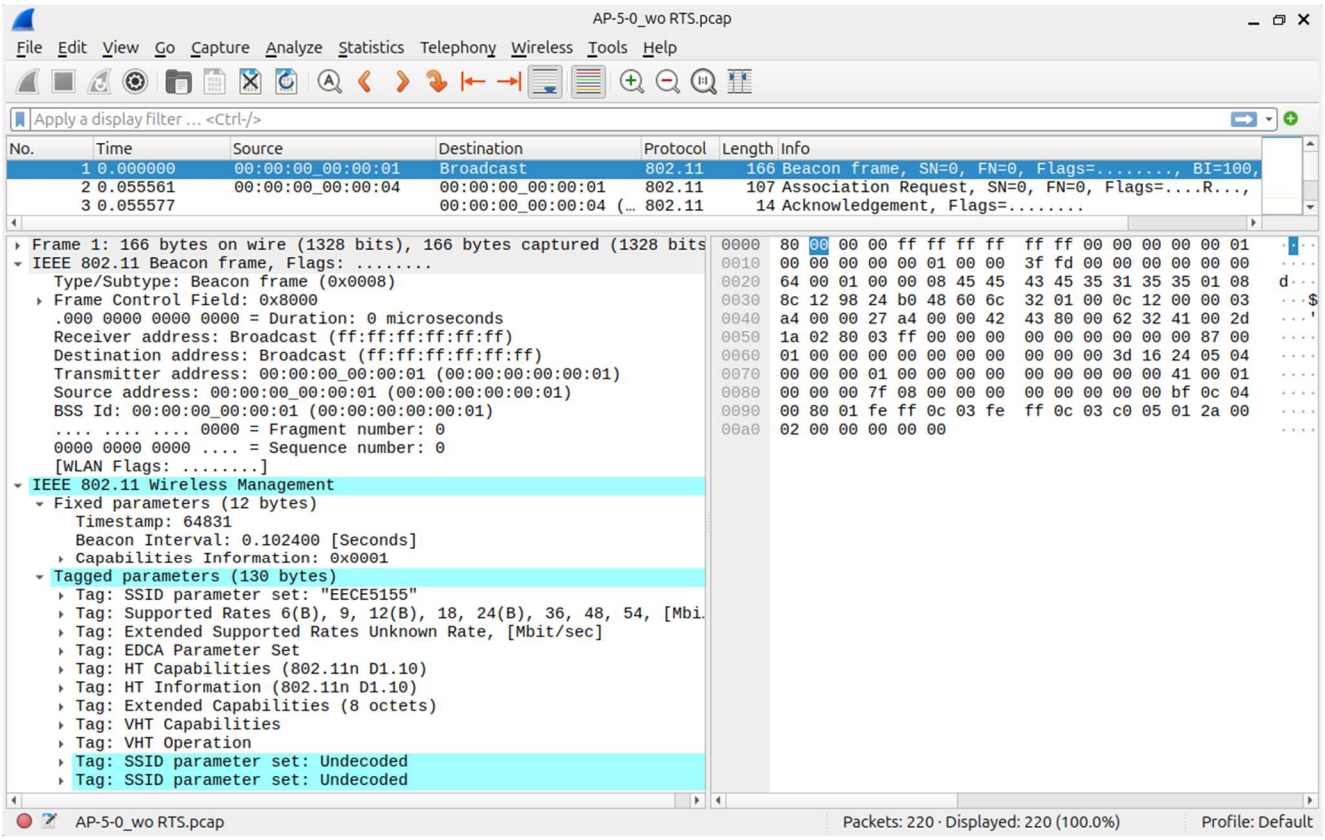
b) Take a look to a beacon frame. Which are the most relevant parameters defined in it?

- In a Wi-Fi network, the beacon frame plays a critical role in managing and organizing network communication. It's broadcasted periodically by Access Point (AP) to announce the presence of the network and to synchronize connected devices. Let's break down the most relevant parameters defined in a beacon frame, as observed in the screenshots you provided.

The top 3 relevant parameters out of many in a Beacon frame are as follows:

1. **Timestamp**:
   - The timestamp is included in each beacon frame to help synchronize the clocks of all devices in the network. This synchronization ensures that all devices operate within the same time frame, allowing for efficient coordination and minimizing overlapping transmissions. In an infrastructure network, the AP uses the timestamp to keep all connected devices synchronized.
2. **SSID (Service Set Identifier):**
   - The SSID is the network name that identifies the wireless network. It's a unique name assigned to the AP, allowing clients to distinguish between different networks in the area.
   - In the screenshots, the SSID would be set by the AP and is visible in the beacon frame so that devices scanning for networks can detect and attempt to connect to this specific network.

3. **Beacon Interval:**
    - This is the interval (typically measured in milliseconds) at which the AP sends out beacon frames. It defines how frequently the AP advertises its presence to surrounding devices.
    - A lower interval may improve connectivity for devices frequently scanning for networks but can increase network overhead.



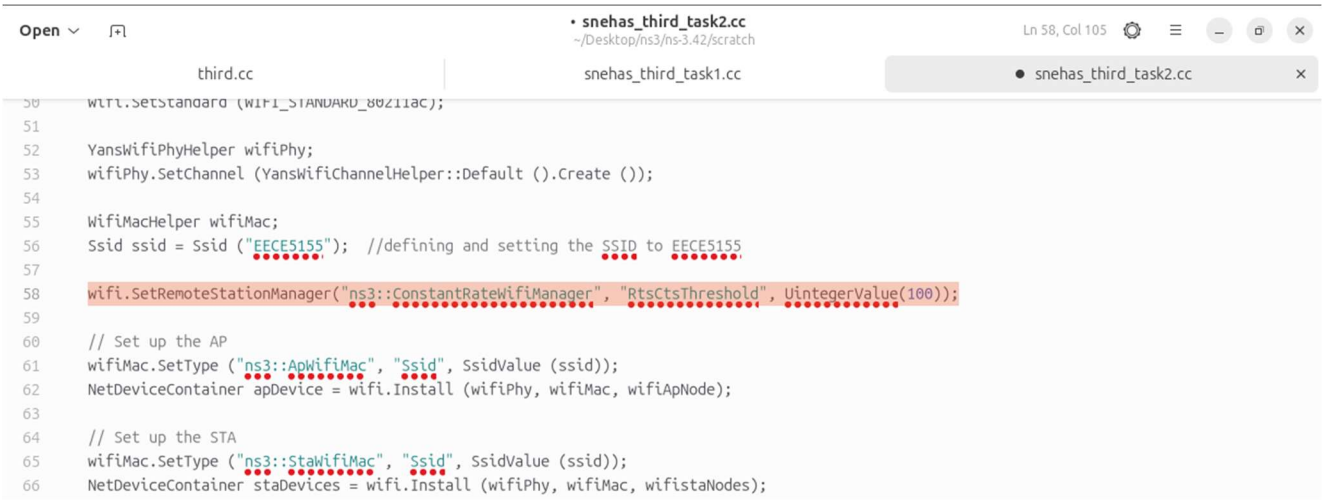
c) Are there any collisions in the network? When are these collisions happening?

- In an infrastructure Wi-Fi network, collisions can occur due to multiple devices attempting to access the shared wireless medium simultaneously. To identify and understand collisions in this context, let's go through the signs of collisions, the typical conditions under which they happen, and how to detect them in your network data.
- In a Wi-Fi infrastructure network, every unicast data frame sent to or from the Access Point (AP) should receive an ACK. If there is no ACK following a data frame within a short time frame, it's likely due to a collision or interference that prevented the frame from reaching its destination.
- Filter for Missing ACKs for data frames without corresponding ACKs by searching for patterns where a data frame is sent but no ACK follows. This pattern often suggests that the data frame encountered a collision or other issues.
- Collisions often occur when multiple devices attempt to send data simultaneously to the AP or when the hidden node problem is present (devices out of each other's range but both within range of the AP).
- **Duplicate or closely timed frames**: Collisions are more likely to happen during periods of high activity, especially if multiple clients are trying to send or receive data shortly after a beacon frame (when devices are "awake" and ready to communicate). If you notice frames being retransmitted or missing ACKs right after beacon intervals, it might indicate that several clients are attempting to communicate immediately after each beacon, leading to collisions.

However, in our pcap, we don't see much retransmissions or missing ACKs. Therefore, we can conclude that there are no collisions in this transmission.

d) As in Task 1, force the utilization of the RTS/CTS handshaking process and repeat the simulation. Are there any collisions in data frames? Explain why.

Like Task 1, we incorporate the Remote Station Setter in our code at line #58 as highlighted and shown in the screenshot attached below:



- This ensures that every data frame, regardless of size, will use the RTS/CTS handshake, reserving the channel before transmission. This setup helps mitigate collisions, especially in dense environments or those with hidden nodes.

- After adding this line, we rerun the simulation to observe the effects of the RTS/CTS mechanism. With RTS/CTS forced, we see RTS and CTS frames for every data frame sent. These control frames serve to reserve the channel, preventing other nodes from transmitting during this period. Ideally, each data frame should be followed by an ACK frame from the receiver if it was received successfully. Missing ACKs usually indicate collisions, but with RTS/CTS enabled, these should be minimized.

Observed Results and Explanation

- After enabling RTS/CTS:
  - **Collisions in Data Frames**: With RTS/CTS, there should be significantly fewer or no collisions in data frames. The RTS/CTS mechanism reserves the channel before each transmission, preventing other nodes from accessing the channel and reducing the chances of collisions.

  - **Why Collisions are Reduced**: The RTS/CTS handshake works as a channel reservation method. When a node sends an RTS, other nodes refrain from transmitting until the CTS and data frame exchange is complete. This coordination minimizes the chances of simultaneous transmissions, especially helpful in environments with hidden nodes or dense client configurations.

Conclusion

- Enabling RTS/CTS in your infrastructure network should lead to:
  - A reduction in data frame collisions due to controlled channel access.

  - Fewer retransmissions, as the handshake minimizes the need for resending packets.

By forcing RTS/CTS, the network achieves more efficient and collision-free communication, although at the cost of added overhead from the control frames (RTS and CTS) in each transmission. This setup is ideal for collision-prone environments or networks with hidden nodes.

C. Lessons Learnt

In task 2, we learn insights for designing and managing Wi-Fi networks, especially in high-density or hidden-node environments, where collision avoidance and efficient resource management are essential for stable performance. Here are the lessons learned from Task 2 regarding the use of infrastructure mode in Wi-Fi networks and the impact of RTS/CTS handshaking:

1. **Understanding Infrastructure Mode and the Role of the Access Point (AP)**
    a. **Centralized Communication:** In infrastructure mode, the Access Point (AP) serves as the central communication hub. All devices (stations or STAs) communicate through the AP, which manages network traffic, handles packet forwarding, and acts as the coordinator.
    b. **Network Coordination:** The AP regularly broadcasts beacon frames, which help clients synchronize, discover the network, and understand its capabilities (like supported data rates and encryption).
    c. **Collision Potential:** Even in infrastructure mode, collisions can happen when multiple devices attempt to send data to the AP simultaneously. This is especially true in high-density networks or when hidden nodes are present.

2. **The Role and Benefits of RTS/CTS Handshaking**
    a. **Purpose of RTS/CTS:** The RTS/CTS mechanism is designed to prevent collisions by reserving the channel before data transmission. RTS (Request to Send) and CTS (Clear to Send) frames establish a “reservation” for the channel, making it less likely that other devices will transmit at the same time.
    b. **Collision Reduction with RTS/CTS:** When RTS/CTS is enabled (by setting RtsCtsThreshold to zero), the network sees fewer collisions and retransmissions. This is because each transmission is prefaced by a handshake that reduces the likelihood of simultaneous access to the channel.
    c. **Trade-Off:** While RTS/CTS reduces collisions, it adds overhead to each transmission, slightly lowering the effective data rate. This trade-off is necessary in dense or hidden-node-prone networks to maintain stable communication, though it may not be beneficial in low-density networks where collisions are rare.

3. **Interpreting Network Performance Through Packet Analysis**
    a. **Analyzing Beacon Frames:** Beacon frames contain important network information, such as SSID, supported rates, and synchronization information. Observing these frames in Wireshark helps in understanding the setup and health of the network.
    b. **Retransmissions as Indicators of Network Health:** Frequent retransmissions indicate potential network congestion or collision issues. Analyzing these retransmissions provides insights into whether RTS/CTS or other congestion control methods are needed.
    c. **Monitoring Data and ACK Frames:** Observing the sequence of data and ACK frames helps identify missing acknowledgments, which can suggest points of failure in communication and possible collision points.

In summary for Task 2, we learned the following:

- **Infrastructure Mode Operation:** How the AP functions as the network's central hub, broadcasting beacons and handling communications from multiple STAs.
- **Collision Detection and Mitigation:** Techniques to identify and understand collisions in infrastructure mode and how RTS/CTS can help.
- **RTS/CTS Trade-Offs:** While RTS/CTS improves reliability by reducing collisions, it introduces overhead. In practice, using an adaptive RTS/CTS threshold helps balance performance and reliability.
- **Packet Analysis Skills:** Leveraging tools like Wireshark to understand network behavior through retransmissions, missing ACKs, and beacon frames, enabling better troubleshooting and network optimization.