Code explain:

* RTS threshold:
  + is a **packet size limit** that determines **when** the RTS/CTS mechanism is used.
  + **Packets larger than rtsThreshold** → RTS/CTS is used.
  + **Packets smaller than rtsThreshold** → Sent directly without RTS/CTS.
* Sets up how Wi-Fi nodes **control their transmission rates** using the **ConstantRateWifiManager**.
  + wifi.SetRemoteStationManager(...): Configures how **Wi-Fi nodes choose their transmission rate**.
  + "ns3::ConstantRateWifiManager": Uses a **fixed data rate**, meaning the nodes do not dynamically adjust their rate based on network conditions.
  + "DataMode", StringValue("ErpOfdmRate24Mbps"): Sets the **data transmission rate** to **24 Mbps** using **ERP-OFDM modulation**
  + "ControlMode", StringValue("ErpOfdmRate24Mbps"): Sets the **control frame transmission rate** to **24 Mbps**
  + This code **forces all transmissions** to use **24 Mbps** (no automatic rate selection).

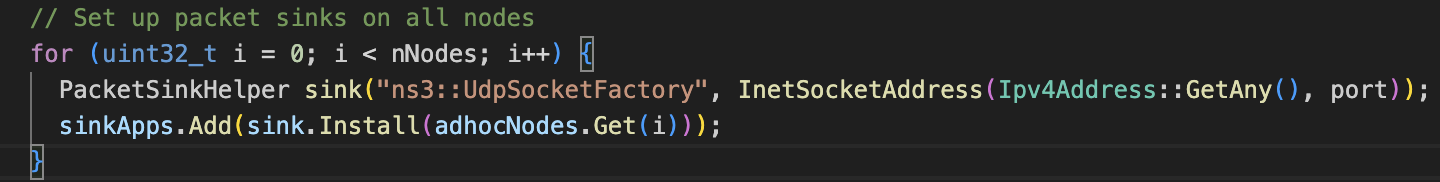
A black background with text

Description automatically generated

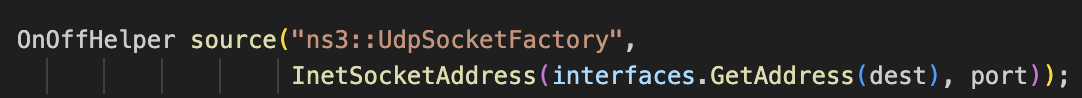
A black background with white text

Description automatically generated

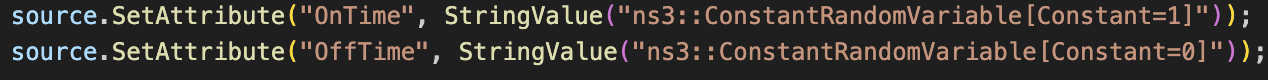
* + port = 9: Specifies the **UDP port** on which the packet sink will listen.
  + sinkApps: Stores **packet sink applications** (which receive packets)
  + sourceApps: Stores **sending applications** (configured later in the code).
* This loop **installs a packet sink** (receiver) on **every node** in the network. This means **every node can receive UDP packets**, making it possible for any node to act as a **destination**.



* + PacketSinkHelper sink(...): Creates a **UDP packet sink** (a receiver that listens for packets).
  + "ns3::UdpSocketFactory": Specifies that **UDP** is used (instead of TCP).
  + InetSocketAddress(Ipv4Address::GetAny(), port): Listens for packets on **any available IP address** at **port 9**.
  + sink.Install(adhocNodes.Get(i)): Installs the **sink application** on node i.
  + sinkApps.Add(...): Stores the **installed sink applications** for future reference.
* This section of the code configures **UDP traffic sources** where each node sends packets to a randomly selected destination node.



* + **OnOffHelper** is used to generate **UDP traffic**
  + "ns3::UdpSocketFactory" specifies that **UDP** is used.
  + InetSocketAddress(interfaces.GetAddress(dest), port)



* + **OnTime = 1s** → The source **remains ON (sending packets) continuously**.
  + **OffTime = 0s** → The source **never turns OFF** (i.e., there are no idle periods).
  + Effectively, each node **keeps sending packets continuously**.

## Attributes:

* Packet arrival rate:
  + This range is too broad and should likely be clarified. If the packet interval is 10 packets per second (as mentioned earlier), the arrival rate should be **100ms per packet**. (M xem lại để fit nhé)
* Simulation Timing:
  + The simulation duration range (10s - 100s) is unclear. If the simulation is supposed to evaluate up to 30 nodes, the duration should be consistent across all scenarios.

A graph of different types of data

Description automatically generated with medium confidence

**Aggregate Throughput vs. Node Count (Top Left - Blue)**

Aggregate throughput measures the total amount of data successfully received by all nodes per unit time.

**Received Bytes**: Total bytes received successfully across all flows.

**Simulation Time**: Total duration of the simulation (in seconds).

**Increasing Trend**: As the number of nodes increases, aggregate throughput generally increases.  
**Fluctuations**: Some dips in throughput occur due to packet collisions and retransmissions.  
**Higher Node Count → Higher Throughput**: Since more nodes are transmitting, the total traffic increases.

**Why?**

* More nodes generate more data, increasing **total network throughput**.
* **However**, increased contention for the channel leads to some inefficiencies.

**Per-Node Throughput vs. Node Count (Top Right - Green)**

Measures the **average** throughput per individual node in the network.

**Declining Trend**: Per-node throughput **decreases** as more nodes join.  
**High Variance**: Throughput is unstable due to competition for the channel.

**Why?**

* More nodes **share the same bandwidth**, so each node gets **less throughput**.
* **CSMA/CA collision avoidance** causes delays when multiple nodes try to transmit.
* **Without RTS/CTS**, collisions are more frequent → more retransmissions.

**Packet Delivery Ratio (PDR) vs. Node Count (Bottom Left - Red)**

PDR represents the ratio of successfully received packets to the total transmitted packets.

Received packets: The total number of packets sucessfully received.

Sent packets: The total number of packets transmitted.

**Declining Trend**: As node count increases, PDR **drops significantly**.  
**Initially High PDR** (For small networks, PDR is close to 1).  
**High Packet Loss** for large node counts.

**Why?**

* **CSMA/CA increases contention** as nodes increase → More **collisions**.
* **No RTS/CTS → More Hidden Node Collisions**:
  + When nodes cannot sense each other’s transmissions, packets collide at the receiver.
* **Queue Overflow**: Nodes may drop packets if they cannot access the channel in time.

**End-to-End Delay vs. Node Count (Bottom Right - Purple)**

End-to-end delay is the average time taken for packets to travel from source to destination.

Total Delay: The sum of delays for all receiving packets.

Received Packets: The total number of packets sucessfully received.

**Low Delay Initially**: When there are few nodes, packets are delivered quickly.  
**Sharp Increase After ~25 Nodes**: Delay spikes dramatically at **high node counts**.

**Why?**

* **More nodes → More contention → More backoff delays**.
* **Retransmissions due to collisions increase delay**.
* **Traffic congestion causes queueing delays**.
* **Without RTS/CTS**, more packet collisions occur → More retransmission delays.

## CONCLUSION

As a result, the CSMA/CA protocol without RTS/CTS in wireless networks faces significant challenges when network density increases. The lack of collision avoidance mechanisms leads to frequent packet collisions, reducing throughput and increasing delays. The hidden node problem exacerbates transmission failures, while the exposed node issue causes unnecessary transmission deferrals, further limiting efficiency. As contention grows, the reliance on exponential backoff results in unpredictable delays and reduced fairness among nodes.