**There are two parts in this document:  
1. My response to your question.**

**2. My question.**

1. “I have checked your codes, according to our project requirement, the bus node should send the ‘raw payload’ data. but in your development, the bus node send the Ethernet data that contains the ‘raw paylod’.”

This is what you asked. But I have no idea the reason why bus node sends the “raw payload”. Your goal is to send bus msg containing specific information to gwTSNBus, isn’t it?

This is required by our client, raw data should be sent.

2. When I set the center frequency to 60GHz, wireless nodes can’t communicate with each other wireless ones. I think that is due to band configuration.

Even if I set the center frequency to 60GHz using hybridNetScalarRadio module, the band is still set to 2.4GHz. In addition, there are only 7 bands – “2.4GHz”, “5GHz”, “5GHz(20MHz)”, “5GHz(40MHz)”, “5GHz(80MHz)”, “5GHz(160MHz)” and “5.9GHz” (referring to "inet/physicallayer/wireless/ieee80211/mode/Ieee80211Band.h", and “inet.physicallayer.wireless.ieee80211.packetlevel.Ieee80211TransmitterBase.ned”).

Please let me know whether my opinion is right. And if not, how can I solve this problem?

I do not know the details of this question. I have searched online, but I am not sure if the following idea works or not.

To modify the center frequency of wireless nodes to 60GHz, and considering the fact that INET framework by default does not directly support configurations for 60GHz, you need to make some custom configurations. Here's a detailed guide in English, along with reference examples to help you make the modifications:

1. \*\*Define the 60GHz Band\*\*: You must first define a 60GHz frequency band in your model. This usually involves extending or modifying the enumeration or class in the `Ieee80211Band.h` file to add a new entry representing the 60GHz band.

2. \*\*Create or Modify the Radio Model\*\*: Since `hybridNetScalarRadio` is your custom radio model, you need to ensure it can handle signals at 60GHz. This might mean modifying its internal logic to correctly transmit and receive signals on the 60GHz band.

3. \*\*Configure the Propagation Model\*\*: At 60GHz frequency, the signal propagation characteristics are significantly different from lower frequencies (like 2.4GHz or 5GHz). You might need to select or develop a signal propagation model suitable for the 60GHz band. This involves modifying your radio environment (usually the `RadioMedium` module) configuration to use the appropriate propagation model.

4. \*\*Adjust Path Loss Model and Other Relevant Settings\*\*: High-frequency signals, such as those at 60GHz, experience more severe path loss and atmospheric absorption. You might need to adjust or customize the path loss model to accurately reflect these conditions.

5. \*\*Update INI Configuration Files\*\*: In your `.ini` configuration files, make sure to specify the new band, modified radio model, and any relevant models and parameters. For instance, ensure the `radioMedium` module uses your custom settings.

These steps require a deep understanding of the INET framework and C++, as you may need to write or modify source code. Moreover, testing and verifying that your configuration performs as expected in simulations is crucial, especially considering the unique propagation characteristics of the 60GHz band.

Please note, the steps described here are conceptual guidelines, and the specific implementation details will depend on your specific needs and the version of the INET framework. If you are not familiar with these advanced customizations, I recommend consulting the INET documentation and source code or seeking help in relevant communities and forums.

***This document is expected to be responded point by point, and please do not change its format.***

|  |
| --- |
| For omnetpp.ini, I have the following questions:  Q1-1: As shown below, Is it possible that this configuration method is not used? The configuration in xml needs to be placed in omnetpp.ini.  \*.configurator.config = xmldoc("config.xml")  Q1-2: You should simplify omnetpp.ini and remove irrelevant content.  Q1-3: As shown the green-marked parts, the “startTime” is not allowed. You cannot configure the start time of the bus node.    Q1-4: Similarly, “initialProductionOffset” cannot be used. |
| A1-1: |
| A1-2: |
| A1-3: |
| A1-4: |

|  |
| --- |
| For omnetpp.ini, I have the following questions:  Q2: In omnetpp.ini, could you please explain the following two contents? i.e., line 42, and line 43. Why you configure these two? |
| A2: |

|  |
| --- |
| For omnetpp.ini, I have the following questions:  Q3-1: What is the unit of “qos”?  "BsrcBus\_Node[2]dstipwirelessHost6dstmacgwTSNBuspri3qos20000"  Is it mean qos is 20000 ns?  Q3-2: I have one thing need to confirm with you, qos is 20000, then the transmission interval should be same, i.e., the transmission interval from “gwTSNbus” to destination should be 20000. And why you configure the interval “20ms”?  “gwTSNBus” will get the qos value from the bus message, and “gwTSNBus” will use this value to set the transmission interval of a flow. The transmission interval cannot set by you. |
| A3-1: |
| A3-2: |

|  |
| --- |
| For omnetpp.ini, I have the following questions:  Q4-1: No input should be configured for the onlineConfig. You should fix it.    Q4-2: If I delete all the JSON files in the simulations folder, will your program still run normally?  No JSON should be contained, you should fix it. |
| A4-1 |
| A4-2 |

|  |
| --- |
| For omnetpp.ini, I have the following questions:  Q5: Is it possible to integrate all settings related “wilressHost”, for example, use “\*.wirelessHost{0..50}.app[0].typename” to replace “\*.wirelessHost{0..1}.app[0].typename” |
| A5 |

|  |
| --- |
| For omnetpp.ini, I have the following questions:  Q6: “MyUdpSinkApp” cannot be used as the application name for “gwTSNBus” and “gwTSNad”. You should changed the name as below, just change the name is okay.  \*.gwTSNBus.app[0].typename = "**gwTSNBusApp**"  \*.gwTSNad.app[0].typename = "**gwTSNadApp**" |
| A6 |

|  |
| --- |
| Q8-1: The format of shared memory is wrong. It is different from the project requirements.  The value of type is not standard, only “bus”, “tsnsw”, “ap”, “wirelessnode”, “gatewayTSNBus”, and “gatewayTSNWireless”, are accepted, you should change them:   1. If a node is “BusNode”, then its type is “bus”; 2. If a node is “TSN\_backboneSW”, then its type is “tsnsw”; 3. If a node is “TSSwitch” , then its type is “tsnsw”; 4. If a node is “wireless host”, then its type is “wirelessnode”. 5. If a node is “gwTSNBus”, then its type is “gatewayTSNBus” 6. If a node is “gwTSNad”, then its type is “gatewayTSNWireless”. |
| A7 |

|  |
| --- |
| Q8: Users will test the project.  In the first step, the user will randomly change the NED file to change the topology, such as changing the number of TSN nodes, changing the number of wireless nodes, AP nodes, etc., or changing the connection relationship between nodes.  In the second step, the user will adjust the configuration information of omnetpp.ini to adapt to the NED file.  Step 3: Run the simulation  The fourth step is to obtain the simulation results.  You need to evaluate, would your code pass the test if users followed the steps above? If the following conditions are met, the test can pass:  1. Your code should run without errors.  2. The simulation results can be obtained after the run is completed.  3. The end-to-end delay is less than the QOS delay constraint, the jitter is very small, the packet loss rate is 0, and the bit error rate is 0. |
| A8 |