in folder "lorawan/examples/"
add file "class-c-end-device-example.cc"
Reference --> https://github.com/QiuYukang/lorawan/blob/class-c/examples/class-c-end-device-example.cc

in folder "lorawan/helper"
1. file lorawan/helper/lorawan-mac-helper.h
=== add contributor info, (line 19-20), it help us to have similar line number
 *
 * Modified by: QiuYukang <b612n@qq.com>

=====add header in line 31
#include "ns3/class-c-end-device-lorawan-mac.h"

====line 46 change
from enum DeviceType { GW, ED_A };
to   enum DeviceType { GW, ED_A, ED_C };

=== add codes line 91 -101
 /**
   * qiuyukang add 2019.12.07
   * Set the duty cycle of EU region.
   *
   * \param dutyCycle the duty cycle limitation(0.0~1.0, default 0.01 for EU region).
   */
  void SetDutyCycle(double dutyCycle);

====add codes line 137 to 140
 /**
   * Perform region-specific configurations for the 868 MHz EU band.
   */
  void ConfigureForEuRegion (Ptr<ClassCEndDeviceLorawanMac> edMac) const;

==== add codes line 174 to 177
 /** * * * *
 * Modified by: QiuYukang <b612n@qq.com>
 * Modified by: QiuYukang <b612n@qq.com>
 * Modified by: QiuYukang <b612n@qq.com>
 * Modified by: QiuYukang <b612n@qq.com>
   * Perform region-specific configurations for the ALOHA band.
   */
  void ConfigureForAlohaRegion (Ptr<ClassCEndDeviceLorawanMac> edMac) const;

===== add code 194 to 195
  // qiuyukang add 2019.12.07
  double m_dutyCycle; //!< The dutyCycle limitation for EU region.

Reference --> https://github.com/QiuYukang/lorawan/blob/a8a6cf64de262a2d6af6d1b623807bbbc970e4d2/helper/lorawan-mac-helper.h

2. File "lorawan/helper/lorawan-mac-helper.cc"

==== line 19-20, add contributor info
```
 *
 * Modified by: QiuYukang <b612n@qq.com>
```

=== line 35, change
from LorawanMacHelper::LorawanMacHelper () : m_region
(LorawanMacHelper::EU)
to LorawanMacHelper::LorawanMacHelper () : m_region
(LorawanMacHelper::EU), m_dutyCycle (0.01)

=========line 57 to 59, add codes
```
case ED_C:
    m_mac.SetTypeId ("ns3::ClassCEndDeviceLorawanMac");
    break;
```

========= line 78 to 90, add codes
```
// qiuyukang add 2019.12.07
void
LorawanMacHelper::SetDutyCycle (double dutyCycle)
{
  if (dutyCycle > 0.0 && dutyCycle <= 1.0)
    {
      m_dutyCycle = dutyCycle;
    }
  else
    {
      NS_LOG_WARN ("SetDutyCycle error! Duty cycle must be a double between 0
and 1.");
    }
}
```

===== add codes line no. 103 to 106
```
else if (m_deviceType == ED_C && m_addrGen != 0)
    {
      mac->GetObject<ClassCEndDeviceLorawanMac> ()->SetDeviceAddress
(m_addrGen->NextAddress ());
    }
```

==========add codes line 133 to 154
```
else if (m_deviceType == ED_C)
    {
      Ptr<ClassCEndDeviceLorawanMac> edMac = mac-
>GetObject<ClassCEndDeviceLorawanMac> ();
      switch (m_region)
```

```
      {
      case LorawanMacHelper::EU:
       {
         ConfigureForEuRegion (edMac);
         break;
       }
      case LorawanMacHelper::ALOHA:
       {
         ConfigureForAlohaRegion (edMac);
         break;
       }
      default:
       {
         NS_LOG_ERROR ("This region isn't supported yet!");
         break;
       }
      }
  }

===== add codes, line 218 to 253
void
LorawanMacHelper::ConfigureForAlohaRegion
(Ptr<ClassCEndDeviceLorawanMac> edMac) const
{
  NS_LOG_FUNCTION_NOARGS ();

  ApplyCommonAlohaConfigurations (edMac);

  ////////////////////////////////////////////////////
  // TxPower -> Transmission power in dBm conversion //
  ////////////////////////////////////////////////////
  edMac->SetTxDbmForTxPower (std::vector<double>{16, 14, 12, 10, 8, 6, 4, 2});

  ////////////////////////////////////////////////////////////
  // Matrix to know which DataRate the GW will respond with //
  ////////////////////////////////////////////////////////////
  LorawanMac::ReplyDataRateMatrix matrix = {{{{0, 0, 0, 0, 0, 0}},
                                        {{1, 0, 0, 0, 0, 0}},
                                        {{2, 1, 0, 0, 0, 0}},
                                        {{3, 2, 1, 0, 0, 0}},
                                        {{4, 3, 2, 1, 0, 0}},
                                        {{5, 4, 3, 2, 1, 0}},
                                        {{6, 5, 4, 3, 2, 1}},
                                        {{7, 6, 5, 4, 3, 2}}}};
  edMac->SetReplyDataRateMatrix (matrix);

  ////////////////////
  // Preamble length //
  ////////////////////
  edMac->SetNPreambleSymbols (8);
```

```
  ////////////////////////////////////
  // Second receive window parameters //
  ////////////////////////////////////
  edMac->SetSecondReceiveWindowDataRate (0);
  edMac->SetSecondReceiveWindowFrequency (869.525);
}

=====add codes line 352-387
void
LorawanMacHelper::ConfigureForEuRegion (Ptr<ClassCEndDeviceLorawanMac>
edMac) const
{
  NS_LOG_FUNCTION_NOARGS ();

  ApplyCommonEuConfigurations (edMac);

  ////////////////////////////////////////////////
  // TxPower -> Transmission power in dBm conversion //
  ////////////////////////////////////////////////
  edMac->SetTxDbmForTxPower (std::vector<double>{16, 14, 12, 10, 8, 6, 4, 2});

  ////////////////////////////////////////////////////////
  // Matrix to know which DataRate the GW will respond with //
  ////////////////////////////////////////////////////////
  LorawanMac::ReplyDataRateMatrix matrix = {{{{0, 0, 0, 0, 0, 0}},
                                        {{1, 0, 0, 0, 0, 0}},
                                        {{2, 1, 0, 0, 0, 0}},
                                        {{3, 2, 1, 0, 0, 0}},
                                        {{4, 3, 2, 1, 0, 0}},
                                        {{5, 4, 3, 2, 1, 0}},
                                        {{6, 5, 4, 3, 2, 1}},
                                        {{7, 6, 5, 4, 3, 2}}}};
  edMac->SetReplyDataRateMatrix (matrix);

  ////////////////////
  // Preamble length //
  ////////////////////
  edMac->SetNPreambleSymbols (8);

  ////////////////////////////////////
  // Second receive window parameters //
  ////////////////////////////////////
  edMac->SetSecondReceiveWindowDataRate (0);
  edMac->SetSecondReceiveWindowFrequency (869.525);
}

======modify line 437-442
 // channelHelper.AddSubBand (868, 868.6, 0.01, 14);
 // channelHelper.AddSubBand (868.7, 869.2, 0.001, 14);
```

```
 // channelHelper.AddSubBand (869.4, 869.65, 0.1, 27);
  channelHelper.AddSubBand (868, 868.6, m_dutyCycle, 14);
  channelHelper.AddSubBand (868.7, 869.2, m_dutyCycle, 14);
  channelHelper.AddSubBand (869.4, 869.65, m_dutyCycle, 27);
```

===== change line 590-591
from
```
Ptr<ClassAEndDeviceLorawanMac> mac =
        loraNetDevice->GetMac ()->GetObject<ClassAEndDeviceLorawanMac> ();
```
to (become 1 line -->590)
```
Ptr<EndDeviceLorawanMac> mac = loraNetDevice->GetMac ()-
>GetObject<EndDeviceLorawanMac> ();
```

=== change 748-749
from
```
Ptr<ClassAEndDeviceLorawanMac> mac =
        loraNetDevice->GetMac ()->GetObject<ClassAEndDeviceLorawanMac> ();
```
to (become 1 line --> 748

Reference --> https://github.com/QiuYukang/lorawan/blob/a8a6cf64de262a2d6af6d1b623807bbbc970e4d2/helper/lorawan-mac-helper.cc

in folder "lorawan/model/"
1.create file lorawan/model/class-c-end-device-lorawan-mac.h
Reference --> https://github.com/QiuYukang/lorawan/blob/class-c/model/class-c-end-device-lorawan-mac.h

2.create file lorawan/model/class-c-end-device-lorawan-mac.cc
Refrence --> https://github.com/QiuYukang/lorawan/blob/class-c/model/class-c-end-device-lorawan-mac.cc

2a. (the next step - (step no 2.a) is add 2 line of code in file "lorawan/wscript"
==line 25
```
'model/class-c-end-device-lorawan-mac.cc',
```

===line 87
```
 'model/class-c-end-device-lorawan-mac.h',
```

3. disable "private" in line 434 file lorawan/model/end-device-lorawan-mac.h
//private:
Reference --> https://github.com/QiuYukang/lorawan/blob/class-c/model/end-device-lorawan-mac.h

4. add a line of code in line no. 548 file " lorawan/model/end-device-lorawan-mac.cc"
```
Time channelWaitTime = m_channelHelper.GetWaitingTime (logicalChannel);
```
Refrence --> https://github.com/QiuYukang/lorawan/blob/class-c/model/end-device-lorawan-mac.cc

5. Changes the code in File "lorawan/model/end-device-status.h"

=====Changes code line 187-188, file "lorawan/model/end-device-status.h"
from
```
  EndDeviceStatus (LoraDeviceAddress endDeviceAddress,
                   Ptr<ClassAEndDeviceLorawanMac> endDeviceMac);
```
to
```
 EndDeviceStatus (LoraDeviceAddress endDeviceAddress,
                  Ptr<EndDeviceLorawanMac> endDeviceMac);
```

==== line no. 259
from Ptr<ClassAEndDeviceLorawanMac> GetMac (void);
to Ptr<EndDeviceLorawanMac> GetMac (void);

==== line no. 329
from Ptr<ClassAEndDeviceLorawanMac> m_mac;
to Ptr<EndDeviceLorawanMac> m_mac;

Reference --> https://github.com/QiuYukang/lorawan/blob/class-c/model/end-device-status.h

6. Change the code in file "lorawan/model/end-device-status.cc "
=====line 50, 51
from
```
EndDeviceStatus::EndDeviceStatus (LoraDeviceAddress endDeviceAddress,
                                  Ptr<ClassAEndDeviceLorawanMac> endDeviceMac)
```
to
```
EndDeviceStatus::EndDeviceStatus (LoraDeviceAddress endDeviceAddress,
                                  Ptr<EndDeviceLorawanMac> endDeviceMac)
```

======line 175
from Ptr<ClassAEndDeviceLorawanMac>
to   Ptr<EndDeviceLorawanMac>

Refrence --> https://github.com/QiuYukang/lorawan/blob/class-c/model/end-device-status.cc

7. add code in line no 62-65, file "lorawan/model/network-scheduler.h "
```
 /**
   * Send data to the end device in the specified recieve window
   */
  void DoSend(Ptr<Packet> data, LoraDeviceAddress deviceAddress, int window);
```

Reference --> https://github.com/QiuYukang/lorawan/blob/class-c/model/network-scheduler.h

8. add code from line 134 to 180 lorawan/model/network-scheduler.cc

```
void
NetworkScheduler::DoSend(Ptr<Packet> data, LoraDeviceAddress deviceAddress,
int window)
{
  NS_LOG_FUNCTION(this << data << deviceAddress << window);

  // Broadcast frame
  if (deviceAddress.IsBroadcast ())
    {
      // Find all avalible gateways to send broadcast
      std::list<Address> gwAddresses = m_status->GetAvalibleGatewaysForBroadcast
();
      if (gwAddresses.size() == 0)
      {
        NS_LOG_INFO ("No avalible gateways for broadcast!");
        return;
      }

      // Create a broadcast frame
      Ptr<Packet> packet = m_status->CreateBroadcastPacket (data);

      // Send the packet through all avalible gateways
      for (auto it = gwAddresses.begin(); it != gwAddresses.end(); it++)
        {
          NS_LOG_DEBUG ("Send a broadcast frame through gateway " << *it << " .");
          m_status->SendThroughGateway (packet, *it);
        }
    }
  // Unicast frame
  else
    {
      // Check whether we can send a reply to the device, again by using
      // NetworkStatus
      Address gwAddress = m_status->GetBestGatewayForDevice (deviceAddress,
window);

      NS_LOG_DEBUG ("Found available gateway with address: " << gwAddress);
      if(gwAddress == Address())
      {
        NS_LOG_DEBUG ("No suitable gateway found.");
        return;
      }

      // Create a frame
      Ptr<Packet> packet = m_status->GetDataPacketForDevice(data, deviceAddress,
window);

      // Send the reply through that gateway
```

```
      m_status->SendThroughGateway (packet, gwAddress);
    }
}
```

Reference --> https://github.com/QiuYukang/lorawan/blob/class-c/model/network-scheduler.cc

9. Add code in line 100-105 file lorawan/model/network-server.h

```
  /**
   * Send a packet to a end device
   * \param data payload in frame
   * \param deviceAddress address of end device
   */
  void Send(Ptr<Packet> data, LoraDeviceAddress deviceAddress);
```

refernce --> https://github.com/QiuYukang/lorawan/blob/class-c/model/network-server.h

10. Changes the code in file "lorawan/model/network-server.cc"
===line 145 & 146
from

```
  Ptr<ClassAEndDeviceLorawanMac> edLorawanMac =
    loraNetDevice->GetMac ()->GetObject<ClassAEndDeviceLorawanMac> ();
```

to

```
  Ptr<EndDeviceLorawanMac> edLorawanMac =
    loraNetDevice->GetMac ()->GetObject<EndDeviceLorawanMac> ();
```

==== add code in line 184 to 189

```
void NetworkServer::Send(Ptr<Packet> data, LoraDeviceAddress deviceAddress)
{
  NS_LOG_FUNCTION(this << data << deviceAddress);

  m_scheduler->DoSend(data, deviceAddress, 2);
}
```

Reference --> https://github.com/QiuYukang/lorawan/blob/class-c/model/network-server.cc

11. file "lorawan/model/network-status.h "
=== line 27 add header

```
#include "ns3/class-c-end-device-lorawan-mac.h"
```

====line 57 changes
from `void AddNode (Ptr<ClassAEndDeviceLorawanMac> edMac);`

to void AddNode (Ptr<EndDeviceLorawanMac> edMac);

=====line 89 to 99 add

```
 /**
   * Get all addresses of gateway which is available for transmission
   */
  std::list<Address> GetAvalibleGatewaysForBroadcast ();

  /**
   * Create a broadcast packet for end devices
   *
   * \param data payload for broadcast frame
   */
  Ptr<Packet> CreateBroadcastPacket (Ptr<Packet> data);
```

=====line 115 to 118 add

```
  /**
   * Get the data packet for the specified device address.
   */
  Ptr<Packet> GetDataPacketForDevice (Ptr<Packet> data, LoraDeviceAddress
edAddress, int windowNumber);
```

Reference --> https://github.com/QiuYukang/lorawan/blob/class-c/model/network-status.h

12. file " lorawan/model/network-status.cc"

==== line 60 change the code

from NetworkStatus::AddNode (Ptr<ClassAEndDeviceLorawanMac> edMac)

to NetworkStatus::AddNode (Ptr<EndDeviceLorawanMac> edMac)

=====line 69 & 70

from

```
Ptr<EndDeviceStatus> edStatus = CreateObject<EndDeviceStatus>
      (edAddress, edMac->GetObject<ClassAEndDeviceLorawanMac>());
```

to

====add codes line 168 to 186

```
std::list<Address>
NetworkStatus::GetAvalibleGatewaysForBroadcast ()
{
  NS_LOG_FUNCTION (this);

  std::list<Address> gwAddress = std::list<Address>();
```

```
  std::map<Address, Ptr<GatewayStatus>>::iterator iter;
  for (auto &iter : m_gatewayStatuses)
    {
      // fixme: this frequency should be set according to different regions, not fixed at
869.525 (ED region)
      if (iter.second->IsAvailableForTransmission (869.525))
        {
          gwAddress.push_back (iter.first);
        }
    }

  return gwAddress;
}



=======188 to 223

Ptr<Packet>
NetworkStatus::CreateBroadcastPacket (Ptr<Packet> data)
{
  LoraDeviceAddress address;
  address.SetNwkID (0x7F);
  address.SetNwkAddr (0x1FFFFFF);

  LorawanMacHeader mHdr;
  mHdr.SetMType (LorawanMacHeader::UNCONFIRMED_DATA_DOWN);
  mHdr.SetMajor (1);

  LoraFrameHeader fHdr;
  fHdr.SetAsDownlink ();
  fHdr.SetFPort (1);  // TODO Use an appropriate frame port based on the application
  fHdr.SetAddress (address);
  fHdr.SetAdr (false);
  fHdr.SetAck (false);
  fHdr.SetAdrAckReq (0);
  fHdr.SetFCnt (0);
  fHdr.SetFPending (false);

  Ptr<Packet> packet = data->Copy();

  // Add packet tag
  LoraTag tag;
  // fixme: this frequency should be set according to different regions, not fixed at
869.525 (EU region)
  tag.SetFrequency (869.525);
  // fixme: this DataRate should be set according to different regions, not fixed at DR0
(EU region)
  tag.SetDataRate (0);
```

```
  packet->AddHeader (fHdr);
  packet->AddHeader (mHdr);
  packet->AddPacketTag (tag);

  return packet;
}


=========add codes 241 to 246
Ptr<ClassAEndDeviceLorawanMac> classAMac =
     edStatus->GetMac ()->GetObject<ClassAEndDeviceLorawanMac> ();
  Ptr<ClassCEndDeviceLorawanMac> classCMac =
     edStatus->GetMac ()->GetObject<ClassCEndDeviceLorawanMac> ();
  NS_LOG_DEBUG ("Class_A_Mac:" << classAMac << "  Class_C_Mac:" <<
classCMac);
  NS_ASSERT (classAMac != 0 || classCMac != 0);


=========changes code in 248 to 260
from (line 248 - 260)

// Apply the appropriate tag
  LoraTag tag;
  switch (windowNumber)
    {
    case 1:
      tag.SetDataRate (edStatus->GetMac ()->GetFirstReceiveWindowDataRate ());
      tag.SetFrequency (edStatus->GetFirstReceiveWindowFrequency ());
      break;
    case 2:
      tag.SetDataRate (edStatus->GetMac ()->GetSecondReceiveWindowDataRate
());
      tag.SetFrequency (edStatus->GetSecondReceiveWindowFrequency ());
      break;
    }

to (248-274)

  // Apply the appropriate tag
  LoraTag tag;
  switch (windowNumber)
    {
    case 1:
      if (classAMac != 0)
        {
          tag.SetDataRate (classAMac->GetFirstReceiveWindowDataRate ());
        }
      else if (classCMac != 0)
        {
          tag.SetDataRate (classCMac->GetFirstReceiveWindowDataRate ());
```

```
      }
     tag.SetFrequency (edStatus->GetFirstReceiveWindowFrequency ());
     break;
    case 2:
     if (classAMac != 0)
      {
       tag.SetDataRate (classAMac->GetSecondReceiveWindowDataRate ());
      }
     else if (classCMac != 0)
      {
       tag.SetDataRate (classCMac->GetSecondReceiveWindowDataRate ());
      }
     tag.SetFrequency (edStatus->GetSecondReceiveWindowFrequency ());
     break;
    }


===== add 280 to 324

Ptr<Packet>
NetworkStatus::GetDataPacketForDevice (Ptr<Packet> data, LoraDeviceAddress
edAddress, int windowNumber)
{
  // Get the reply packet
  Ptr<EndDeviceStatus> edStatus = m_endDeviceStatuses.find (edAddress)-
>second;
  edStatus->SetReplyPayload(data);
  Ptr<Packet> packet = edStatus->GetCompleteReplyPacket ();
  Ptr<ClassAEndDeviceLorawanMac> classAMac =
     edStatus->GetMac ()->GetObject<ClassAEndDeviceLorawanMac> ();
  Ptr<ClassCEndDeviceLorawanMac> classCMac =
     edStatus->GetMac ()->GetObject<ClassCEndDeviceLorawanMac> ();
  NS_LOG_DEBUG ("Class_A_Mac:" << classAMac << "  Class_C_Mac:" <<
classCMac);
  NS_ASSERT (classAMac != 0 || classCMac != 0);

  // Apply the appropriate tag
  LoraTag tag;
  switch (windowNumber)
   {
    case 1:
     if (classAMac != 0)
      {
       tag.SetDataRate (classAMac->GetFirstReceiveWindowDataRate ());
      }
     else if (classCMac != 0)
      {
       tag.SetDataRate (classCMac->GetFirstReceiveWindowDataRate ());
      }
     tag.SetFrequency (edStatus->GetFirstReceiveWindowFrequency ());
```

```
      break;
    case 2:
      if (classAMac != 0)
        {
          tag.SetDataRate (classAMac->GetSecondReceiveWindowDataRate ());
        }
      else if (classCMac != 0)
        {
          tag.SetDataRate (classCMac->GetSecondReceiveWindowDataRate ());
        }
      tag.SetFrequency (edStatus->GetSecondReceiveWindowFrequency ());
      break;
    }

  packet->AddPacketTag (tag);
  return packet;
}
```

Reference ---> https://github.com/QiuYukang/lorawan/blob/class-c/model/network-status.cc

13. Add codes line 129 to 132 file "lorawan/model/lora-device-address.h"

```
 /**
   * Determine if an address is a broadcast address.
   */
  bool IsBroadcast ();
```

Reference --> https://github.com/QiuYukang/lorawan/blob/a8a6cf64de262a2d6af6d1b623807bbbc970e4d2/model/lora-device-address.h

14. add codes 165 to 169 file " lorawan/model/lora-device-address.cc"

```
bool
LoraDeviceAddress::IsBroadcast ()
{
  return m_nwkId.Get () == 0x7F && m_nwkAddr.Get () == 0x1FFFFFF;
}
```

Reference --> https://github.com/QiuYukang/lorawan/blob/a8a6cf64de262a2d6af6d1b623807bbbc970e4d2/model/lora-device-address.cc#L166

RUN

```
./waf configure
./waf
```